HACKING MATERIALS

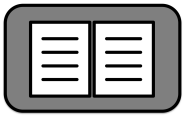
Group Handbook

A guide to help you start your scientific adventures

update: 8/09/22

hackingmaterials.lbl.gov

maintained by Anubhav Jain



(best viewed in 2-page mode)

Table of Contents

Thank you! **122**

Preamble

"Organization is a means of multiplying the strength of an individual".
- Peter Drucker

Welcome to the HackingMaterials research group! The purpose of this guide is to ease your transition into our research group and help make your time here as productive and enjoyable as possible.

An editable copy of this guide, where *everyone* is welcome to make comments and suggestions, is always available at:
https://bit.ly/2huxUJW

You can also download the latest "PDF release" of this guide at
https://hackingmaterials.lbl.gov/handbook.pdf

The page size of this document is A5, which makes it work well for **two-page** viewing and printing. We suggest you give that a try.

For additional tips and tricks, please see the Hackbook (internal document) on the group Google Drive.

We thank Valve software (a video game company) for openly publishing their employee handbook, which helped inspire this effort. Also, thank you to those who help create and update this guide; each of us gains from the contributions of those before us, and we hope you are able to pay it forward to future members by contributing as well.

About our group

Our group is located at Lawrence Berkeley National Laboratory (LBNL) in Berkeley, California. LBNL is managed by the University of California at Berkeley, which is located just down the hill. 13 Nobel prizes have been awarded to scientists from LBNL. The lab has spectacular views of San Francisco, which is located across the bay and is about a 30 minute drive or BART train ride away. Berkeley itself is a vibrant city of 115,000 people filled with cafes, restaurants of all types, and cultural activities.

Our group aims to tackle some of the most important problems lying at the intersection of materials science and computer science. We differ from a traditional materials theory group in our emphasis on building long-term software, in leveraging large supercomputers, and in applying statistical learning to materials problems. Most of our projects use a "materials genomics" approach, a new mode of research that has

tremendous potential to discover new materials and to improve our fundamental understanding of how materials behave.

Our research group is its own microcosm within the materials science theory efforts at Berkeley. In the same way that Berkeley is a small city adjacent to the bigger city of San Francisco, our group is a smaller unit linked to the larger theory groups of Kristin Persson, Gerbrand Ceder, Jeffrey Neaton, and Mark Asta – creating a close-knit community of materials theory within Berkeley. We also collaborate with groups external to the Berkeley area, and thus it is almost always the case that *someone* within our collaboration circle has experience with any new methods or applications you might be interested in. We hope you are able to leverage many of these resources during your stay!

Many new discoveries remain to be uncovered in the field of materials design and in our relatively new subfield of materials informatics. Your contributions are urgently needed to make this new vision a reality - welcome!

Before you arrive

Although many things can only be taken care of after arriving at LBNL, here are a few simple things you should do in advance.

Join the Slack group!

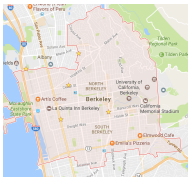
The berkeleytheory Slack group spans several research groups at LBNL. There are channels to ask housing questions, programming questions, science discussion, and general hijinx.

How? **Contact Anubhav - he'll introduce you.**

Order a computer!

Postdocs and graduate students - let's order your workstation in advance so that it's ready by the time you arrive.

How? **See Appendix on purchasing a computer.**



Find a place to live!

Berkeley and the surrounding areas are wonderful places to live, but finding an apartment can be difficult and is best done with knowledge of the various neighborhoods.

How? **See Appendix on housing.**

After arriving at LBNL

Welcome to Berkeley! Here are a few pointers for getting started.

Getting set up to work

Here is a checklist of things you should complete within your first week:

- ❏ **Set up your computer**. See Appendix documentation on how to do this and some recommended software to install.
- ❏ **Request to be added to the group email list.** Alex Dunn can add you to *hackmat@lists.lbl.gov*

- ❑ **Complete the checklist that HR gives you in your welcome package.** Note: postdocs cannot select retirement plans; an obligatory UCRS DCP is deducted from your pre-tax salary each month but the money belongs to you.
- ❑ **Complete all LBNL training courses.**
- ❑ **Set up the employee wifi**. Note that the visitor wifi is open access. To connect to employee wifi, go to ***https://software.lbl.gov/***, search for "Wireless Networking", and download the configuration file.
- ❑ **Request access to the group's Google Drive folder.**
- ❑ **Install VPN for connecting to the lab network from home.** For example, this lets you download research articles from home**.** See ***https://software.lbl.gov*** for instructions on installation.
- ❑ **Obtain user accounts for any computing resources you may be using**. See documentation later in this handbook.
- ❑ **Schedule a 30-minute biweekly meeting time with Anubhav.** Ask about any other meetings.
- ❑ **Obtain a license for any software packages you might be using.** For example, you may need to be added to the VASP users list (for VASP, you should also register for the forum.)
- ❑ **Set up the printer.** The printer is located in the corner of the third floor of building 62 near the water fountain. Follow instructions at ***http://bit.ly/2sB6yIX*** to connect. Note: if you are having issues connecting to the printer, ensure during setup that the connection protocol being used is Line Printing Daemon - LDP. Mac may default to IPP instead which will not work. The name of the printer is *ms-div-copyprnt3.lbl.gov*.
- ❑ **Request after-hours access**. By default, you will not have off-hours site access to building 62, *i.e.*, on weekends, holidays,

and from ~6pm to 7am on weekdays. To obtain off-hours site access, email the ESDR admin (*esdradmin@lbl.gov*), tell them you need after-hours building access, and cc Anubhav.

- ❑ **Ask to read the proposal that funds your work.** This will help explain the impact of your project, the long-term plans and goals, and how your project fits in with other efforts.
- ❑ **Have your picture taken for the group website.** Coordinate a time with Anubhav, who will take the picture.
- ❑ **Update your name tag outside your office.** Contact Charlotte Standish for help if needed.
- ❑ **Get a key to your office.** See ***http://bit.ly/2vGGWe9***
- ❑ **Add your name to the 62-203 mail room.** Note that interns and short-term visitors do not need to do this.
- ❑ **Say "hi" to your neighbors!** Working here will be more pleasant if you get to know some of the people around you. One good time to introduce yourself is when you see people eating lunch in the kitchen area.
- ❑ **Send Anubhav an email confirming you have completed everything on this list.**

Getting situated in your office

There is no rule that says your office must be dull and generic. Decorate your lab space with photos, posters, or other personal touches. Take a moment to make this space your own!

Food and coffee

There is a common kitchen in building 62 with microwave, fridge, and coffee machine. I don't know who the coffee machine belongs to but you

can try to make friends with whomever that is. There is a coffee machine in the 2nd floor of Molecular Foundry ($1/cup or $15/month) that some of the other postdocs in the group can tell you more details about. For *emergency* coffee situations, Anubhav has an espresso machine in his office that you can borrow (but wait for the go-ahead to come in, since he is often in the middle of a videoconference).

There are no vending machines in building 62, but if you go down to the first floor you walk to building 66 and there are some vending machines on the bottom floor. There are also some vending machines on the bottom floor of 67 (Molecular Foundry).

The only real food is in the lab cafe, which is about a 15 minute walk from our building. The menu is posted at (some items change weekly): ***http://www.bayviewcafelbl.com***

Mail and fax

The incoming and outgoing mail corridor is located in building 62, second floor. You can find an empty slot and put your name there; sometimes, your mail will end up there, other times your mail will just end up in a common pile, so you may need to check both. Please do not have any personal (non-business related) mail sent to your LBNL address - this is not allowed.

If you would like to mail something internally (including the benefits office, which is not on the hill), first get an envelope either from 62-309 or from building 66 (room 237-250, make a left right after entering). Scratch off all the previous mail stops and write down the destination mail stop (something like: 90P-0101) and then put it in the outgoing

mailbox in the mail corridor in 62. There are also some miscellaneous mailing supplies in 62-309 on the shelves.

There is a fax machine in 62-309. For outside numbers, dial 9 first, then the country code (1), then the rest of the number.

Equipment and conference rooms

Should you need it, Anubhav has a projector that you can borrow. If he is not in the office, feel free to go in and just grab it (it is visible on the gray shelf). Just leave a note and remember to return it.

Anyone can reserve conference rooms through the LBNL Google calendar. Make sure you are logged in to your lbl.gov account.To book a room, add an event to your own Google Calendar (first) and use the option within Google calendar to add a room (in the detailed options for the calendar event). You will see a list of LBNL rooms displayed. *e.g.*, 62-203 (big main room) and 62-253 (smaller room with poor wi-fi). Note that when adding a room, you will see a number in parenthesis like (20) - that is approximately the number of people that the room can accommodate, giving you a sense of the room size. To see the availability of one or more conference rooms over time for planning purposes, note that each conference room has its own Google calendar. Just add the conference room's calendar to your list of calendars (find the area that says "Other calendars" and then type the room into "Add coworker's calendar"). For more detailed information, see ***https://commons.lbl.gov/display/fac/Conference+Rooms***.

The Panic Monster

The Panic Monster is a red doll in Anubhav's office based on a blog post from *Wait But Why*. If you see the Panic Monster on Anubhav's desk, it is best not to bother him. If you see the Panic Monster on *your* desk, it means you have lots of work to catch up on and you need to get working! If you do happen to get a visit from the panic monster, fret not; a day or two of dedicated catch-up work is usually enough to chase him off.

Postdoc union

Note that Berkeley postdocs have unionized with the *International Union, United Automobile, Aerospace and Agricultural Implement Workers of America* to obtain collective bargaining agreements. Joining the union is an option, and many of the details are present here: ***http://uaw5810.org*** You can also message the Slack group to get opinions from the current postdocs in the group.

Postdoc resources

Berkeley Lab offers many resources for postdoc. See for example:

http://postdocresources.lbl.gov
https://postdoc.berkeley.edu/resources

Vacation days

You will receive a set number of vacation / personal time off (PTO) days that will be outlined in your hiring package. For union postdocs, the union has currently negotiated 24 PTO days per year (in addition to standard lab holidays) along with other benefits.

You should coordinate the specific days of vacation and personal time off with Anubhav, especially for an extended absence.

What to do if you're sick

If you're sick, ***do not come to the office***. This is very important; otherwise, you can get others sick and potentially bring down the productivity of the entire group. Instead, work from home or take a sick day to rest, relax, and recover. Simply e-mail Anubhav and let him know what you're doing. Just don't come into the office!

Filling out your timecard (LETS)

Every month, you are required to fill out your timecard at ***https://lets.lbl.gov***. This is mostly straightforward but here are a few pointers:

- To fill out your work hours, leave "Earning Type" as Regular and "Shift" as 1. For project id, activity id, and days, use the information Anubhav gives you. You do not need to fill out the work/job number or the specific days on which you worked.
- For sick or vacation days, set "Earnings Type" to the appropriate value. You will need to enter both the number of days as well as

the specific dates that you took sick leave/vacation. If you took vacation or sick days, simply deduct those number of days from your project (proportionally if you have multiple projects).
- In total, the number of days should match the "Work days" listed in the top-left. Note that if it is your first or last month in the group, this might not be true if you join/leave mid-month.
- When finished, click "Run Report", then "Release".
- Note that your "Leave Balance" in the bottom-left assumes 8 hours per day.

Miscellaneous administrative issues

There are now links on how to navigate various administrative items (e.g., requesting a key to your office, requesting conference travel, etc.) at: ***http://bit.ly/2vGGWe9***. You may need to be logged in (and perhaps on LBNL network or VPN) to access it.

Other issues

If you are struggling with stress or other personal problems, you can contact the LBNL Employee Assistance Program (EAP), which provides free and confidential counseling, consultation, and referral for LBNL staff. If you are comfortable doing so, you can also discuss the problem with Anubhav to brainstorm if there are ways forward.

Foreign travel

In many respects, LBL is similar to academic institutions. However, when it comes to foreign travel (non-vacation), it is quite different. There are very strict procedures if you wish to work on projects /

conduct professional activities while not in the U.S., whether that is for an international conference or hoping to continue working remotely while in a foreign country. If you are planning on traveling to a foreign country for anything other than vacation, please schedule a meeting with Anubhav as soon as you can about it and we can discuss. Note that the issue of foreign travel has come up several times in the past and has led to quite major problems when not handled correctly.

Places to work outside of your office

Anubhav is much more interested in your research output than where you work. Indeed, one of the advantages of choosing computational science as a career is that it can afford you some more flexibility location-wise than other jobs. Overall, you are encouraged to work where you feel best *from time-to-time* to maximize your energy and productivity. The more accurate policy is that the more productive you are in terms of output (see "questions for self assessment" section), the less Anubhav cares about where you are doing your work.
If you do end up spending a day working elsewhere, follow these rules:

1. Let me know when in advance and also your collaborators
2. Get your work done
3. Be available, e.g., don't miss meetings and keep an eye on your email
4. Overcommunicate (i.e., send me an email with your progress for the day)

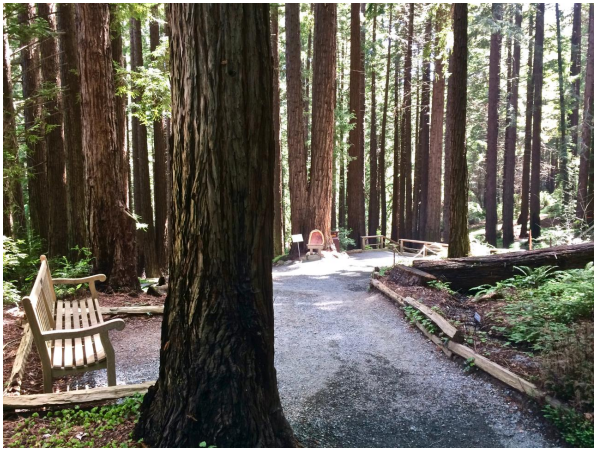
When those rules are followed (particularly the last one), there is usually no problem in working outside your office from time to time. Indeed, this handbook even tells you where you might go.

Some places to work apart from your office include:

- The Molecular Foundry 3rd floor lounge
- The LBNL coffee shop and library reading room (downstairs from the cafeteria cash register area)
- The UC Botanical Garden and Redwood Grove (free admission for LBNL)
- UC Berkeley campus, including the Free Speech Movement cafe which has rotating newspaper headlines
- Downtown and campus-area coffee shops

The Molecular Foundry 3rd floor lounge has a view overlooking San Francisco.

The library reading room near the lab cafe is a quiet and attractive place to work.



The UC Botanical Garden Redwood Grove is a 5-minute walk from our offices.

Telework

It can happen that you are required to work from home for a long period without a possibility to visit the main site. You probably can move for that period somewhere else inside the US (international telework is highly likely not an option, talk to Anubhav first). If it happens, here is a list of things that may be useful for you:

- ❏ Check time zones - all your work still happens on Berkeley time - and update your LBNL google calendar with time when you may be unavailable.
- ❏ Make sure that you have a comfortable working place at home - ergonomics is very important as you will move less than before. If you need something from your office you can check if you can take it home with you.
- ❏ Make sure that you have a cozy working space - you should feel nice there and want to come back every day. Hygge concepts may help.
- ❏ Try to organize a working schedule with time tracking apps like Pomodoro or Toggle and virtual task boards like Trello or Miro.
- ❏ Maybe it is time to start using the Bullet Journal Method?
- ❏ Podcasts and blogs can give you some tips on work from home.

Making purchases

You are encouraged to make purchases that are likely to save you a lot of time. For example, if a commercial version of a software is superior to open-source alternatives, then you should purchase the commercial version. Your time is valuable and if we can solve a problem with funds, then we should try to do so.

Purchases are usually paid for through **project ids** that Anubhav can provide you with. For items less than $100, you should initiate purchases on your own provided that you know the correct project id (just let Anubhav know afterward so he is aware of the charge). For items greater than $100, contact Anubhav first.

There are also a couple of important things to know about purchases that may be different than a university department:

- It can be very difficult if not impossible to "pay first, get reimbursed later" if you don't have the prior pre-authorization. For example, if your lab computer breaks, don't just take it to the Apple store, pay out of pocket for a $300 repair, and then expect the lab will reimburse you later. Similarly, don't make a purchase with personal funds and expect the lab can authorize you for that purchase later. If you have questions, contact Anubhav.
- Note that you are also not allowed to make any commitments or changes to a purchase request on the lab's behalf. For example, if you've already gotten authorization for a purchase but then need to change the shipping address, need to change the date the item is received, or make any further changes, it's actually the lab's "buyer" that needs to negotiate that. This is all a little complicated; you can refer to ***https://bit.ly/2ZEXEwq*** or talk to Anubhav if confused.

The procedure for making purchases depends on the purchase type:

- **Software**: Many popular commercial software libraries (e.g., Microsoft Office) can be purchased through ***software.lbl.gov***. At

check out, the approver for the purchase is Amapola Comayas or Brendon Smith.

- **Office supplies, computer accessories**: Check the LBNL's Ebuy (*not* Ebay) first via ***procurement.lbl.gov***. If the item or an equivalent is available, this is the easiest way to make the purchase (for both you and LBNL administration). At check out, the approver for the purchase is Amapola Comayas or Brendon Smith. Otherwise, see below.
- **Books**: First, see "Resources for learning new topics". After that, if you'd still like to purchase a book, first check if the book is available on Ebuy - this is the simplest purchase option. If it's not available on Ebuy or it is much more expensive on Ebuy than from another seller, follow the instructions below for "other purchases".
- **Other purchases:** For other purchases, please review the information and fill out the form here (you may need to be logged into your LBNL Google account): ***http://bit.ly/2kX41ZQ***

Conference travel

It is important to be connected to the research community. Thus, our group has a policy that all postdocs and grad students are **required** to attend at *minimum* 2 conferences per year (more is fine). If it is your first year in the group, you can simply attend the conferences and listen to talks. After your first year, you are expected to be presenting talks or posters at conferences. This will ensure that:

- you keep up to date on developments in the field
- you will get to know the *people* in the field

- you are broadcasting your work to the research community. Many if not most people learn about new research by hearing about it at a conference. Thus, if you want people to know about your work, you must be willing to tell people about it.

You should identify conferences you'd like to attend several months (usually ~4 months, perhaps ~6 months for international travel) in advance. Usually, this is around the same time that abstract deadlines are due.

Once you have identified a conference you'd like to attend, please take the following actions:

- Tell Anubhav about the conference and what project you'd like to present
- If this is foreign travel, you need to be ***very careful***. DOE has placed many restrictions on foreign travel, and if you do not follow all the guidelines you will ***not*** be reimbursed!
- **As soon as possible** - submit a conference travel request form. This form is a very basic (i.e., 2 minutes to fill out) - find the appropriate approval form on the esdradmin web site: ***http://bit.ly/2vGGWe9***
    - Note that our group's travel arranger is Charlotte Standish
    - If you do not submit the travel request form several months in advance, **you may not receive LBNL approval to attend.** This is especially the case for any travel outside the U.S. or larger conferences like MRS.
- If you haven't done so already, make sure your travel profile (*e.g.*, your frequent flier programs) are completed for the lab. E-mail ***esdradmin@lbl.gov*** if you don't have one yet.

- Work with Anubhav to submit an abstract. You should send him the proposed abstract (with all details - title, authors, text, figures, etc.) with at least 3 days advance notice.

Note that there is usually a two-step process to approval. The first step is "conference services" approval and the second step is TREX approval. Confusingly, LBNL might send you a "travel approved" email after the first step, with details about the second step hidden only in the body of the email. Make sure you have both approvals before proceeding! If ever in doubt, email ***esdradmin@lbl.gov***

Once you have received approval to attend the conference, please take the following steps:

- Make sure you register for the conference in time to receive any early registration discount *(normally on one's own credit card then reimbursed later)*
- Book a hotel *(normally on one's own credit card then reimbursed later)*
- Book a flight - please do this early to avoid last-minute flight rate spikes. For help with this step, use the "Local travel request", "domestic travel request" or "foreign travel request" on the esdradmin web site: ***http://bit.ly/2vGGWe9*** . You can also book in coordination with Charlotte Standish with LBNL making the booking. This works better if you identify desired flights in advance, otherwise give Charlotte the preferred dates and times. Note that if for any reason you book your own flights, you should be aware of various LBNL policies on flight booking such as preference for domestic carriers. I don't suggest you do this unless you have an extremely good reason.

- If you are planning to combine vacation and travel, make sure to not exceed the amount of personal days allowed by DOE (***https://bit.ly/2X3gxtB***). Note that days spent traveling to and from the conference count as work days.

In terms of travel receipts and reimbursement:

- If you need early reimbursement, please send an email to ***esdradmin@lbl.gov*** to see if you can get registration costs, etc. reimbursed early. Note that this is not the normal procedure but if you have a need, it should be possible to have this arranged.
- If you are traveling with funding through LBNL (i.e., most cases), you do ***not*** need to save receipts for meals. You will receive a per diem instead. You also do not need receipts for taxi rides under $75, although Anubhav usually submits them anyway when he has them. You also do not need to save your actual airplane tickets for lab-purchased airfare, although again Anubhav usually submits these anyway.
- If you are traveling with outside funding (e.g., the conference organizers are going to reimburse you), save all receipts and tickets as they may be needed for reimbursement.
- The proper way to request reimbursements for trips within the US is through the esdradmin site's "Travel:Domestic" tab available at: ***https://bit.ly/2Y91KKl***. If you have trouble, you can email esdradmin@lbl.gov.
- For international trips (including Canada!), you should get in touch with Anubhav and with the ESDR admin person that you work with. That person will provide you a corresponding form and help you through the (more complex) process of international-travel reimbursement, and Anubhav should approve your comments for those forms as they are reviewed

closely. You need to start the process for international travel very early - do not wait.

- Note that you can have your travel reimbursement direct-deposited into your bank account, which saves the hassle of receiving and depositing checks. More information about how to set this up is at: ***https://travel.lbl.gov***
- Things to remember reimbursing:
    - registration costs
    - hotel costs
    - baggage fees
    - Uber / taxi fare
    - rental car
    - any last-minute poster printing
    - flights (only if you booked yourself)

Pro tip: If you want to see the status of your conference requests, log in to this sheet with your LBNL account: ***http://bit.ly/2n6XCe3***
You can filter the sheet to your requests by right-clicking on the name column and choosing the filter option. You should look for the "(ADMINS ONLY) Approval status" column in order to check your status.

Use of Airbnb: It is possible to stay in an airbnb, but the receipts and documentation are more complicated. Therefore it is typically suggested and much simpler to stay in a standard hotel. However for those without fear of extra paperwork:

- The rental agreement will need to be signed by the individual renting the property with no affiliation to the Laboratory, DOE, or the University of California.

- A rental agreement cannot be signed for a property which is owned by a Laboratory or University employee.
- When submitting for reimbursement, please ensure the following documents are attached to the expense report:
    - Rental/lease agreement
    - Payment receipt (must include owner/property manager name/phone, property address, total rent paid, and payment date (excluding deposits)
    - Itemized list of miscellaneous expenses (if applicable)
    - List of those occupying the property and dates
- Travelers will not be reimbursed for non-commercial lodging rates which exceed the published per diem rate (***https://bit.ly/2RMweBF***). In addition to the above, please note the following restrictions:
    - Cannot be submitted for a prepayment (TREX or eRFIC)
    - Deposits and cancellation fees are non-reimbursable
    - Damage to the property is the responsibility of the traveler(s)
- Please note, the Laboratory will not prepay or authorize a pre-payment expense report for non-commercial lodging.

Asking your advisor for research help

“Drop-in” office hours

Anubhav usually holds 1-hour office hours twice a week where you can just drop in anytime and ask anything (no appointment needed).

Biweekly 30-minute checkups

Anubhav will schedule a time to check up with you every other week for 30 minutes (if you are an undergraduate, the schedule may be different). You do not need to present slides or prepare any formal presentation. Some of the things you can do during these checkups:

- mention anything that is impeding your progress (e.g., lack of equipment, lack of response from a collaborator, long queue wait times at supercomputer, etc.)
- report what you worked on the last week and present your goals for the next week, month, or 3 months to confirm that you are on the right track, not repeating previous work, etc.
- introduce a major problem you are facing and that requires a longer, targeted meeting to brainstorm a solution (just present the problem in enough detail)
- solve very small problems, such as getting feedback on 3-4 presentation slides
- request a decision about something

Longer meetings by appointment

Longer, targeted meetings are welcome so long as you have a clear purpose for them. In particular, if a difficult decision needs to be made or you'd like to brainstorm a technical problem, a longer meeting will almost certainly work better than e-mail. To schedule a targeted meeting, you should first e-mail me and tell me why you'd like to meet (or simply describe your need at a 30-minute meeting). If a meeting is in fact the best solution, we will work out a time and place to have the

meeting. You can use Anubhav's public calendar to suggest a few possible times. The best way is to add his calendar to yours through LBNL Google Calendar (see instructions in the section about Booking Conference Rooms). Another way is to access his calendar through this public link:

http://bit.ly/2ncHcPo

Other times

Please prioritize the methods above for meeting with Anubhav in person and do not simply "drop by" Anubhav's office to ask questions at random times unless:

- the matter is urgent or you are really worried about something
- the issue is of a more personal nature

In the above cases, just stop by anytime!

Email help (and general guidance)

"If I had an hour to solve a problem I'd spend 55 minutes thinking about the problem and 5 minutes thinking about solutions.".
- Albert Einstein

Note: Anubhav does not respond to direct messages on Slack - help is through email or meetings only.

From time to time, you will encounter problems, require suggestions, or otherwise need assistance from your advisor. This is normal, and asking

for help is encouraged so long as you have done your best to solve the problem yourself. Sadly, it is all too easy these days to send e-mails without first investigating a problem yourself, and it is important to remember that your advisor gets many dozens of emails per day. Thus, if the question is truly important and difficult, you should take the time to address the following four questions[1] in your email:

❑ **What is the problem?**

Clearly describe the problem, *starting from the beginning.*

❑ **What is the CAUSE of the problem?**

For example, your immediate problem may be that you need more computing time. But the *cause* of your problem is perhaps that you want to determine the best ordering of a disordered compound and that you anticipate that this will require running many calculations. If you are unsure how to answer this question, ask yourself "why" this problem needs to be solved (some companies such as Toyota employ the "5 Whys" principle - i.e., asking "why" 5 consecutive times to get to the root of the problem rather than fix surface issues).

❑ **What are all the possible solutions to the problem?**

You might think this means list a couple of possible solutions. That's ***not*** what this means. This means list ***all*** possible solutions - every single way the problem could be solved. This includes unconventional options, options that you may not know how to implement or think might not work. If you don't have any solution ideas, list all the avenues you tried

[1] These guidelines are adapted from Dale Carnegie.

(e.g., Google search terms) to find one. If you already tried some solutions but they failed, summarize that information here.

❑ **What solution do you suggest?**

Provide your reason for suggesting this solution.

More often than not, taking the time to answer these questions leads to you solving your own problem. In the cases where that is not true, these responses will make brainstorming solutions to your problem more effective and will also allow Anubhav to provide feedback into your process of generating all possible solutions.

Software help groups

If you have problems with software, and in particular the software maintained by our group and our collaborators, you should contact the appropriate help group. The documentation for the software will list what that channel is; if not, try the Github Issues page. If you contact Anubhav, make sure you address the four questions above as well as provide everything needed (files, test code, etc) to quickly reproduce and debug the problem.

Two other ways to get software help that are more self-guided are:

- If you are having trouble using a particular class or function, look for unit tests within the code, which often demonstrate how to use the class or function
- If the class or function has a unique name (e.g., MaterialsProjectCompatibility), another option is to both Google *and* search on github.com for the particular

class/function. The github.com search will often reveal code snippets from users all around the world.

Independent projects

"You'll learn infinitely better and easier and more completely by picking a problem for yourself that you find interesting to fiddle around with, some kind of thing that you heard that you don't understand, or you want to analyze further, or want to do some kind of trick with - that's the best way to learn something".
- Richard Feynman

"...it is certainly all right and potentially very productive just to mess around. Quick uncontrolled experiments are very productive. They are performed just to see if you can make something interesting happen".
- E.O. Wilson

You may have ideas for research that deviate somewhat from the scope of your official work. If you are interested in conducting an independent project or trying a new idea, talk to Anubhav! If it is in the group interest, we can try to make it happen.

The main metric for a successful pitch is having a plan to prototype and test an idea (e.g., by gathering data) in a rapid fashion.

Our computing systems

Our group's main computing resources are:

- NERSC (the LBNL supercomputing center, one of the biggest in the world)
- Lawrencium (our group owns 4 dedicated nodes on this cluster)
- Eagle Computing System at NREL (for projects funded by DOE EERE)
- Argonne Leadership Computing Facility (sometimes)
- Oak Ridge Leadership Computing Facility (sometimes)

At any time, if you feel you are computing-limited, please contact Anubhav so he can work with you on finding solutions.

NERSC

To get started with calculations at NERSC:

1. Ask Anubhav about whether you will be running at NERSC and, if so, under what account / repository to charge.
2. Request a NERSC account through the NERSC homepage (Google "NERSC account request").
3. Someone at NERSC will validate your account and assign you computing hours
4. At this point, you should be able to log in, check CPU-hour balances, etc. at https://iris.nersc.gov and https://my.nersc.gov.
5. In order to log in and run jobs on the various machines at NERSC, review the NERSC documentation.

6. In order to load and submit scripts for various codes (VASP, ABINIT, Quantum Espresso), NERSC has lots of information to help. Try Google, e.g. “NERSC VASP”.
    a. Note that for commercial codes such as VASP, there is an online form that allows you to enter your VASP license, which NERSC will confirm and then allow you access to.
7. Please make a folder inside your project directory and submit all your jobs there as your home folder has only about 40GB of space. For example, for m2439 project, your work folder path should be something like the following:
    a. /global/project/projectdirs/m2439/YOUR_NERSC_USERNAME
8. If you are using custom Python software in your jobs (such as atomate), it is recommended to install it in the dedicated “software” directory. This is because this folder has much faster IO for the compute nodes than the project or home directories and running Python software often requires accessing many files (rather than just a single executable such as VASP). For example, for the m2439 project, you should install compute-critical Python software in:
    a. /global/common/software/m2439/
9. You can also request a database for your project to be hosted on NERSC. Google “MongoDB on NERSC” for instructions.
10. Visit the workshop on Spin https://www.nersc.gov/users/training/events/

Running Jupyter Notebooks on Cori

Jupyter notebooks are quickly becoming an indispensable tool for doing computational science. In some cases, you might want to (or need to) harness NERSC computing power inside of a jupyter notebook. To do this, you can use NERSC's new Jupyterhub system at ***https://jupyter.nersc.gov/***. These notebooks are run on a large memory node of Cori and can also submit jobs to the batch queues (see ***https://docs.nersc.gov/services/jupyter/*** for details). All of your files and the project directory will be accessible from the Jupyterhub, but your conda envs won't be available before you do some configuration.

To set up a conda environment so it is accessible from the Jupyterhub, activate the environment and setup an ipython kernel. To do this, run the command "pip install ipykernel". More info can be found at ***http://bit.ly/2yoKAzB***. If that did not work, you can also manually add your environment. Assuming your environment is my_env in Python 3:

```
source activate my_env
python -m ipykernel install --user --name my_env
--display-name "aj_te (py3)"
```

Lawrencium

Lawrencium is somewhat different than NERSC in that we must maintain our own software environment and pre-installed binaries for common codes are not available. Thus, maintaining the software environment at Lawrencium is a group endeavour. We currently have 4 nodes which means as a group we can submit jobs that require up to 4 nodes and those jobs will have the highest priority to start running. Note that if our purchased nodes are not enough to sustain our

computing needs, it is also possible to pay per CPU-hour on Lawrencium as well as to increase our purchase order to increase our nodes. Ask Anubhav if you think you need this.

The nodes on Lawrencium are not that fast (2.3 GHz) but there is large number of cores (24 per node). So, it is best for codes that parallelize. Also, make sure to do any file-based work on Lawrencium on the /scratch filesystem, which is not purged. Any operation (loading software libraries, output file writing, etc) that occurs on the /home filesystem will be *extremely* slow. So /home is just for archival purposes, not working purposes.

To use Lawrencium resources, first apply for a new account by visiting this link https://it.lbl.gov/resource/hpc/for-users/hpc-documentation/accounts/user-accounts/ and fill out the user agreement form here (https://sites.google.com/a/lbl.gov/high-performance-computing-services-group/useragreement) and then send an email to hpcshelp@lbl.gov to request an account - make sure the email specifically requests access to the "lr_matminer" condo. Once your account is ready and you are able to login, you can see the group softwares that are not available on Lawrencium by default here: /global/common/software/m2439/example_config_cori

ALCF and OLCF

Both ALCF and OLCF are "leadership computing facilities" meaning that they operate some of the fastest computers in the world. The strength of these facilities is that they offer very large amounts of computer time available for users; the weakness is that is much more

difficult to use these computers. Therefore, it is generally only worth using these resources if you have a significant amount of computing to do (*i.e.*, at least 1 million CPU-hours). Contact Anubhav if you think an account on ALCF or OLCF would be useful.

Our software stack

A brief summary of our software stack includes:

- **pymatgen / pymatgen-db** - for representing and analyzing crystal structures, as well as setting up/performing manual calculations
- **FireWorks** - for executing and managing calculation workflows at supercomputing centers
- **custodian** - instead of directly running an executable like VASP, one can wrap the executable in custodian to detect and fix errors
- **atomate** - for quickly defining multiple types of materials science workflows
- **matminer/automatminer** - for large data analysis and visualization

We also heavily use the **Materials Project** database.

To learn how to use the software stack, you can consult the documentation of the individual codebases as well as review the following resources:

- The 2018 Materials Project workshop: ***https://github.com/materialsproject/workshop-2018***

- The 2014 Materials Virtual Lab presentations: ***https://materialsvirtuallab.org/software/***
- The Materials Project YouTube tutorials: ***https://www.youtube.com/user/MaterialsProject***

Resources for learning new topics

Slack

If you have a specific question, sometimes the easiest solution is to post it to the Slack group and crowdsource the answer.

Books: LBNL, UC Berkeley, public libraries, and the "group library"

As an LBNL employee, you can get access to almost any book you'd like using various channels:

- LBNL has its own library, but it is small and unlikely to contain the book you want.
- LBNL employees can borrow books from the UC Berkeley Library collection using your LBNL ID. You can even reserve the book online and have it delivered to the LBNL library office in building 50, saving you a trip down to campus (this is what I do). Log in through ***http://oskicat.berkeley.edu*** via "My Oskicat" and choose the LBNL login option.
- Your local library (e.g., Berkeley Public Library) often participates in Interlibrary loans. For example, the "Link+" system at Berkeley Public Library connects to many other

university libraries in the area. Again, you can have the items delivered to your local library. This service is extremely useful when an item cannot be found at UC Berkeley or if that item has a long waiting list.

- Our group has some technical books that you can loan. See ***Appendix K: Group library***.
- If you are struggling to find a paper (for example, it is a translation of a russian article and was published in a now defunct journal with no online archive in the 1940's) then send a polite email to library@lbl.gov with as many details as possible. Prepare to be impressed.

You can also purchase books with research funds.

Materials Science

"Don't despair of standard dull textbooks. Just close the book once in awhile and think what they just said in your own terms as a revelation of the spirit and wonder of nature".
- Richard Feynman

It can be difficult to find resources that explain concepts in materials science clearly. Often, struggling through multiple attempts to understand a topic using several different resources in a patchwork and non-linear fashion is the only way forward. That said, the resources listed below are particularly helpful.

Density functional theory

For beginners to density functional theory, I would recommend the book "***Density Functional Theory: A Practical Introduction***", which truly achieves what it states by providing physical insights and relevant information rather than just list equations. A copy is available within the group.

If you are interested to explore applications of density functional theory, you might try the E-book from Professor John Kitchin:
https://github.com/jkitchin/dft-book
Note that this book has chosen to use the Atomic Simulation Environment (ASE) to set up simulations rather than the pymatgen code that we prefer, but that is a minor point.

Finally, for specific calculations with VASP, there are resources online from a 2016 workshop conducted at LBNL, including videos and training materials:
http://www.nersc.gov/users/training/events/3-day-vasp-workshop/
http://cms.mpi.univie.ac.at/wiki/index.php/NERSC_Berkeley_2016

General materials science topics

To gain a quick introduction to many topics in materials science, you might try the (horribly-named) web site from the University of Cambridge: Dissemination of IT for the Promotion of Materials Science (DoITPoMS):
https://www.doitpoms.ac.uk
The explanations in this site are very basic, but what they do cover is well-explained and incorporates helpful visuals. Although you won't

ever master a topic from this site, it is often a good starting point that can help you unlock a more intermediate resource.

There are also some nice chapters in the following e-book:
https://en.wikibooks.org/wiki/Introduction_to_Inorganic_Chemistry
For example, Chapter 5 has a nice rundown of common crystal structures.

Online tools

A nice tool for visualizing phonon modes is:
http://henriquemiranda.github.io/phononwebsite/phonon.html

A nice tool for visualizing Brillouin zones is:
https://www.materialscloud.org/work/tools/seekpath
Note, the high-symmetry labels may occasionally differ from those on the Materials Project.

Databases and information resources

The LBNL library maintains a subscription to many tools and databases for materials science such as SpringerMaterials. A list of these is available here:
https://bit.ly/2HCePDQ

Computer programming

Note that there are usually many excellent resources to choose from when learning computer science topics. You usually have the flexibility of choosing to learn from a book, a video series, or even interactive tutorials like ***www.learnpython.org***. Use the list below as potential

starting points, but there exist many other high-quality alternatives you can find on your own and may be even better-suited to your needs.

Python

For pure beginners to Python, you might try the book "***Head First Python***". It is a fun and easy introduction to Python. Beginners that know a little but not a lot of Python can also look at ***"How to Make Mistakes in Python"*** (ebook). For intermediate programmers, you might try ***"20 Python Libraries You Aren't Using (But Should)"*** (ebook). For advanced programmers, you might try "***Expert Python Programming***".

Data mining and Data Analysis

For learning basic data mining libraries (*pandas*, *scikit-learn*) as well as some skills like using *git* and *Github*, you might try the online YouTube videos from Kevin Markham, an educator at Data School. These videos also do a good job of pointing you to supplementary material:
https://www.youtube.com/user/dataschool
https://github.com/justmarkham

You might also try the book "***Python for Data Science For Dummies***" (please note: this is different than "*Data Science for Dummies*").

Data 100/200 offered by UC Berkeley is also a good source to learn almost all skills you need to do data analysis. The lecture records and slides are published online: **https://ds100.org/**

For a more materials-centric view, you can try working your way through the *Machine Learning In Materials tutorial* in the Appendix of this handbook.

A very comprehensive set of suggestions for further resources is listed here:
http://bit.ly/2jHXIVJ

MongoDb

A (now somewhat old, but still clear) resource for beginning to use MongoDb is the "*The Little MongoDB Book*":
https://github.com/karlseguin/the-little-mongodb-book
There is also an extensive library of webinars on MongoDb on their official web site.

Professional skills: writing papers and presenting talks and posters

- If you have only a relatively short time, try this e-book from Nature Publishing Group: ***http://go.nature.com/2opiiQh*** . It is illustrated by Jorge Cham from PhDComics and is packed with good advice.
- I have written more about giving good presentations in Appendix G - use that for additional tips.
- If you have longer, try the book *Trees, Maps, and Theorems* in our group library (Appendix K). It is from the same author as the Nature e-book (Jean Luc Doumont).

Eleven questions for self-assessment

You might be curious as to whether you are on the right track from a professional standpoint. You can ask Anubhav to give you feedback periodically, and you should do this at least every 6 months or so. Here is a cheat sheet of things he considers when thinking about your progress.

1. How self-driven is your work?
    a. I am far ahead of my supervisor in understanding and guiding my project, so it's necessary that I conceive/design/imagine/build most of what I do. i.e., I am given a vague topic to work on by my supervisor and it's my job to determine both the important problems and design the solutions.
    b. I originate maybe 50% of the ideas that I work on; the other 50% are from my supervisor. Or, given a good description of the problem by my supervisor, I figure out the solution largely independently.
    c. Almost all of what I do was sketched out by my advisor, including the problem and the rough solution; my job is to implement those ideas.
2. When I am assigned a task, I usually complete it:
    a. To an even higher quality standard than asked for and/or much quicker than expected
    b. approximately on time and well-tested and robust so that I know that my solution works under diverse situations. I can declare "mission accomplished successfully" the vast majority of the time.
    c. To minimally achieve the original goal, thus often requiring future revision; OR very late; OR usually by

getting someone else to solve most of the hard parts for me

3. Compared to others in the group, I:
    a. help them more than they help me
    b. help them about the same as they help me
    c. help them less than they help me
4. Regarding the relevant scientific literature for my project, I:
    a. regularly impress my supervisor by integrating new and important papers into my research that were not on my supervisor's radar
    b. have about an equal share of papers I receive from my supervisor versus papers I have discovered on my own (and subsequently adapted my research to account for those papers, i.e., have read and understood them)
    c. typically am the recipient of interesting papers to read from my supervisor
5. In the last year, I have attended (if first year in group) or presented a talk or poster (if in group longer than one year) at at:
    a. More than 2 international conferences (not project workshops / meetings)
    b. 2 international conferences
    c. One or zero international conferences
6. In the last 9 months, I have submitted to a journal:
    a. Multiple first-author papers, or one stellar (e.g., quality level of Nature, Science, etc.) first-author paper
    b. One first-author paper, or significant co-author on a stellar (e.g., quality level of Nature, Science, etc.) paper
    c. No first-author papers

7. When I present a draft of a paper and/or presentation to my supervisor, usually I get back:
    a. Minor revisions
    b. Medium level of revision
    c. Major revisions
8. Will my main work serve as a lasting contribution that others will use and refer to in 5-10 years time?
    a. There is a good chance my work will remain important even after 10 years
    b. Probably 5 years, 10 years is a stretch
    c. Honestly, probably not
9. When my supervisor assigns tasks (e.g., at in-person meetings, etc.), I:
    a. complete tasks quickly and efficiently and provide my supervisor with an update
    b. get around to doing almost all tasks eventually
    c. often forget to complete tasks (e.g., forgot to write it down, etc.) and often need to be asked a second time
10. Have you received any external awards or recognition for your work?
    a. yes, in the last year
    b. yes, in the last 3 years OR my work was featured in a news story (LBNL news, DOE news, etc.) in the past year
    c. not in the last 3 years
11. How is your passion and enthusiasm level?
    a. I feel extremely excited and happy about work
    b. About normal
    c. I feel burnt out or demotivated

If your answers are mainly (c), the questionnaire is probably telling you something that you already know -that you should take some time to reflect on your situation. You might also schedule a meeting with Anubhav to discuss things. If you are answering mainly (b), then you are likely doing fine but it may be worth brainstorming if it's possible to move into category (a) for one or more of your responses. Otherwise, continue the great work!

Leaving the group

If you are leaving the group, we wish the best in all your future endeavors! Some things to ensure before leaving:

- Complete and submit any papers that are close to completion
- Transfer any knowledge or code you have to others; train others to the extent possible
- Transfer any data (e.g., stored at NERSC) to others
- Retrieve any information you need from your LBNL computer accounts (email, Google Drive, etc) - your LBNL accounts will not be accessible after you leave. It's possible to have your LBNL account extended, talk to Anubhav or human resources. For example, if you need to continue collaboration with LBNL, an affiliate status might be granted - this retains your LBNL affiliation.
- If you have any subscriptions paid for under your own account via LBNL funds (e.g., MongoDB Atlas, etc) make sure to retrieve the data and cancel those subscription OR transfer the subscription to someone else. Note that LBNL credit cards will

deny charges past the initial approval dates, and this can lead to missed charges and lost data.
- If you have purchased any software subscriptions (e.g., Microsoft Office), note down the license number of those subscriptions so they can be re-used by the next person.
- After doing everything above, clear off your computer so it can be used by someone else (***https://support.apple.com/en-us/HT201065***)
- Make sure that the "Exit Checkout Sheet" is filled in and handed to LBNL. Otherwise, LBNL will follow up to make sure it's complete later on.
- Postdocs, see further instructions below

Specific instructions for postdocs

If you are planning to leave the group as a postdoc, please let Anubhav know as soon as possible. It usually takes 3-4 months to complete a postdoc hire at LBNL, so a standard "2 weeks notice" will mean that your project will likely go vacant for 3 months, impacting the efforts of all your collaborators and colleagues. With advance notice, we can work to make sure that your own future plans are not impacted while also making sure that all your projects and work can transition smoothly.

If you are a postdoc:

- Use the following link to submit a voluntary resignation letter: ***https://bit.ly/2yptUG3***
- If you have PTOs remaining you can set your official last day as a later date than the last day you will be physically present in the lab (work with HR on this)
- If you have any questions about the exit process, you can email Human Resources at ETAHR@lbl.gov.

Group events

Some of the regular things we do as a group are:

- Lunch every two weeks jointly with our group and members of Persson research group
- ~Monthly coffee hour at Molecular Foundry
- Twice-yearly "CodeBusters" events, where we spend 3 days hacking on code together
- Twice-yearly "Work Away From Work" where we can share some interesting space to work that it's not the LBNL main site
- Yearly "Group Day" where we share our work, have discussions, and zoom out on the big picture of our group
- Sporadic events such as group hikes

A hike on the Matt Davis - Steep Ravine trail.

Fun things to do in the area

Make time to explore some of its recreational activities in the Bay Area. Although there are probably hundreds of online and print resources that can help guide you to things to do, here are a few select ones to start you off:

- ❏ UC Botanical Garden (walkable from our office and free for LBNL employees)
- ❏ Berkeley Marina (walking) - either the boardwalk or the Cesar Chavez loop
- ❏ Ohlone Parkway Trail (easy bike)
- ❏ Indian Rock park
- ❏ Berkeley Rose Garden
- ❏ "Off the Grid" food trucks
- ❏ Elmwood shopping area or outdoor Emeryville mall
- ❏ Detour App - guided tours for locals through your phone
- ❏ Berkeley Jazz / Theater
- ❏ UC 50% off performing arts at Zellerbach Hall
- ❏ Tilden State Park, e.g., Lake Anza trail
- ❏ Bike the Golden Gate bridge to Marin (longer bike)
- ❏ Bioluminescent kayaking tour in Point Reyes (pick a night with little moonlight)
- ❏ SF Film Fest
- ❏ SF Symphony: Many (but not all) concerts have student discount - you reserve any seat at a fixed price
- ❏ Baker beach walk up to Golden Gate Bridge ("Batteries to Bluffs" trail)

- ❏ Ice Cream - Mitchell's, Humphrey Slocombe, Bi-rite, Ice Cream Bar (skip the line, go directly to the back bar and order a "New Orleans Hangover" - non-alcoholic)
- ❏ Muir woods redwood forest
- ❏ Drive to Muir Beach Overlook
- ❏ Drive up to Mount Diablo
- ❏ Hike - Stinson Beach / Matt Davis trail
- ❏ Wine country / Sonoma
- ❏ Point Reyes Lighthouse - Elephant Seal season Dec - Feb
- ❏ Route 1, Big Sur (Bixby bridge area), and "18-mile drive".
- ❏ Explore Monterey and Carmel-by-the-Sea
- ❏ Lake Tahoe - skiing in the winter, hiking/biking/cruises/tourism/casinos in summer
- ❏ Visit Yosemite National Park
- ❏ Berkeley Art Studio - enroll in class for a term or visit a single workshop on ceramics, painting, photography and more

Appendix A: Finding a place to live

Note: It is encouraged that readers of this document also contribute to it when they have gained experience with their own housing situation by adding comments or emailing Anubhav with their suggestions.

Temporary housing while finding a place to live

You may consider finding temporary housing, *e.g.*, through Airbnb, for a month or so while finding a more permanent place to live. It is easier to find housing while you are in Berkeley itself; however, depending on the cost of your temporary housing and moving costs, this strategy can be

more expensive. Also please consider the risk that some listings for rent on some websites are fake. In summer there are many offers of sublease which are cheaper than regular offers and allow you to stay for a short period, you can find them in a Facebook community called UC Berkeley Off-Campus Housing.

Resources for finding housing

Unless you are part of a program that assists you with finding housing, you must find a place to live on your own. Some resources for finding housing include:

- ***http://www.zillow.com/***
- ***https://calrentals.housing.berkeley.edu/***
- ***http://sfbay.craigslist.org/search/eby/apa?***
- ***http://csee.lbl.gov/Housing/Other_Housing_Resources.html***
- ***https://facebook.com/groups/ucberkeleyoffcampushousing/***
- ***https://www.sabbaticalhomes.com/***

You can join the LBNL postdoc mailing list (***http://bit.ly/2nAHAXE***). You do not have to be a postdoc to join. Keep an eye on the posts for room/apartment to rent as well as moving sales.

Notes on the Bay Area housing situation

The Bay Area is a very nice place to live, which has the consequence of many people wanting housing here. Thus, one of the few problems with this area is the very high price and competition for housing. Some things you should be aware of:

- prices in the $2000/month range for a very basic apartment are normal - and can easily go up from there. You can also check

your official rent ceiling at: https://www.cityofberkeley.info/RentBoardUnitSearch.aspx

- buildings tend to be older, and amenities like dishwashers and heating/cooling are hard to find unless you pay premium prices.
- it is normal to have a lot of competition for a place such that you must agree to a lease on the spot or risk losing out. Examples:
    - Anubhav thought he had finalized a place to live for his first year in the North Berkeley area, but during the final signing period another bidder put in an offer for $300/month greater and he thus lost the place.
    - One postdoc in the group thought he had finalized a deal for a place to rent but was late to an appointment to meet with the owner and ended up losing the offer on the spot.

You might not expect these kinds of situations unless you are from a similar area like NYC, so please be aware of them.

Commuting

Note that if you use public transportation daily, you should consider signing up for LBL's program which lets you deduct a bus or BART pass as a pre-tax expense. See ***http://www.wageworks.com/*** for more info.

Biking here is common and there are many bike lanes and shared car/bike routes, but you still need to be careful as biking to the lab will mean going through traffic. The LBNL shuttle has bike racks so you can bring your bike up to the lab with you on the shuttle rather than bike uphill.

Note that the Nextbus app and website will give times that the LBL shuttle (and also city buses) are anticipated to arrive at various stops.

If you are in a rush or just need to get around town, Uber and Lyft are apps that can help get your a ride; the fees tend to be pretty low, especially with UberPool if you're not in a hurry.

If you plan to drive, make sure use the feature in Google Maps to estimate commute times at *a specific time of day*. If you are headed in the same direction as traffic to San Francisco, there are very significant delays near rush hours.

General suggestions when evaluating a place to live

- Look for the nearest grocery store
- Look for the nearest pharmacy
- Do a search for restaurants. Often, the density of restaurants in a place will tell you whether there are other things there as well.
- Perhaps do a Google Street View walk-through of the neighborhood
- If you consider Berkeley note that the North to the campus is considered the safest. It is fine in other parts too but please avoid the area of People's park and don't be in the night on University street or Telegraph avenue. Alternatively, the south east area of Berkeley (Elmwood and Rockridge) is a friendly neighborhood and has good transport links to the lab (Rockridge shuttle).

- Do a Google Transit search on how to get to the lab. Note that to get to the lab itself, you cannot take public transportation. Instead, there is a lab shuttle from several spots in downtown Berkeley and near campus, so you might want to gauge how to get to the nearest shuttle stop. Google "LBL shuttle map" to see the locations of the stops.
- Remember that Uber is very convenient in the Berkeley area, so not everything needs to be ideal location-wise if you need to just get somewhere once in a while.

A note about UC Village

Many postdocs, especially those with families, find that UC Village (sponsored housing from UC Berkeley and LBNL) is a nice place to live and also enjoy the community. Anubhav doesn't have any personal experience with UC Village so it is best to research for yourself through a Google search (it does seem very friendly to having children - e.g., a nice park). For grad students the university housing is most likely not an option due to a very limited number of offerings so it is better not to wait in the line but find off-campus housing on the sites listed above.

What are the different neighborhoods like?

If you want to know how specific neighborhoods or cities surrounding Berkeley are like, the best way is to post a question to the *#help* channel on Slack. You can usually get very good and relevant advice.

Good luck!

Appendix B: Purchasing a computer

Most long-term appointments (graduate student, postdoc, staff) will mean purchasing a new computer. Exceptions include:

- If you are joining an existing project for which another staff member is leaving behind a state-of-the-art computer (e.g., 1-2 years old), we may use that one instead of purchasing a new computer.
- Short-term appointments (e.g., internships) will not involve a computer purchase unless otherwise stated - you will instead receive an excellent computer from the group's stock.

Mac, Windows, or Linux?

You should buy a Mac, and probably a Macbook. Although this sounds extreme, and may even induce strong feelings if you are used to a different system, in practice this has never been much of a problem. Note that I am not an Apple fanatic but simply find that these are the best systems for our type of work because they contain many of the advantages of both Linux and Windows systems in a single package.

Why not Windows? Anubhav used Windows for a very long time; it is nice, but a couple of things make it non-optimal for our work. There is no native Terminal, which you will use heavily, and programs like Cygwin are poor substitutes. Certain seemingly minor decisions made by Windows (directory slashes, line endings) are different than those from Linux, making interoperability between Linux/Mac and Windows systems more problematic (e.g., copying files to and from supercomputing centers can require converting line ending format).

Why not Linux? Linux is fine, but Microsoft Office is not available (which is used by us and most of the materials science research world) and OpenOffice is a poor substitute. Certain videoconferencing software doesn't work well with Linux.

How about Mac? I have my complaints about it, as they are catering more to the general consumer and less to developers. Thus, you really need to spend some time setting up your Mac to make it productive for power users (see Appendix C and Appendix D on how to accomplish this). But for the moment it remains a very good compromise between Linux-like and Windows-like and thus forms the basis for our workstations.

Preliminaries

Here is how to purchase a computer at the lab. Before we begin, a few notes:

- In terms of the mechanics of purchasing:
    - use LBNL Ebuy (not Ebay) wherever possible - you need to be on the lab network (onsite via an ethernet cable) or be connected via the VPN
    - use Amazon, etc. to buy various components if not available via EBuy
- The laptop is government property; you are expected to return it to the group when you are done working at LBNL. Note that Mac computers make it very simple to transfer everything over to your next computer.

- You are free to take your laptop home, on trips, etc., unless you are an intern in which case other restrictions may apply from the internship program.
- The lab receives your computer and tags it before sending it over to you.

Selecting a computer, monitor, and accessories

Your computer workstation is one area where you should just get whatever you think will make you most productive and not care about cost. Seriously, just get what is best and do not worry about cost.

For the computer, you should select a Macbook Pro (any screen size) as mentioned above. You can use the Apple website to browse details. Anubhav uses a 13" Macbook Pro. It is powerful enough to do serious work and light/small enough to use on a plane. A 15" Macbook Pro is also a good choice. If you would like to get anything other than a Macbook Pro, talk to Anubhav.

For the monitor, Anubhav uses a single Thunderbolt display but this is no longer available. One option available is the LG 27MU88-W (4K resolution) monitor which is on Ebuy. Note that one big screen is usually better ergonomically than dual monitors, and you can use the "Spaces" feature of Mac OS/X to quickly flip between virtual screens if needed (this is what Anubhav does). Some people also opt for an extra-wide single monitor (e.g. 43").

For accessories, make sure to get:

- An extra charging cable
- An HDMI adapter dongle

- A Time Machine hard disk (for backup), I have currently use the *Western Digital 4GB Passport for Mac*
- A keyboard. I suggest Apple Wireless Keyboard since I like the feel of Mac keys and I also like a consistent feel between my laptop keyboard and my desk keyboard. If you prefer a larger or ergonomic keyboard, you can get that.
- A mouse/trackpad. I suggest Apple Magic Trackpad. Note that I've found that a mouse is better on Windows but a trackpad is better on Mac. The reason is because the Mac OS has really customized a lot of the interface for the trackpad (e.g., gestures). I also value consistency between my laptop and desk workstation. After awhile you get used to doing everything on your trackpad even if you were previously very productive/accurate with a mouse on Windows.
- (optional) A presentation tool, *e.g.*, Logitech R800

Making the purchase

1) Provide all the details of your selections in an email and send to Anubhav. If all looks OK, he will give you a project and activity ID.
2) Go to eBuy, and for items available there, add them to your cart and submit the requisition with the project and activity ID, and SAS approver as Amapola Comayas or Brendon Smith.
3) For items not available on eBuy, contact esdradmin@lbl.gov (and cc Anubhav) to obtain a procurement form. Fill it out with item details (Vendor, website, price, etc.) and send it back to her.
4) If you select the overnight shipping option (ask Anubhav about this and the related extra costs) most parts, except the computer,

will arrive within a week to 10 days. The computer needs to be tagged by the lab, so with overnight shipping, it should arrive within 2 weeks. Ideally, you will select your computer well before arriving at the lab and won't need overnight shipping.

5) Note that receiving the laptop can take up to 6 weeks from the order date.
6) To track your purchase, contact an administration assistant in the energy technologies area via <esdradmin@lbl.gov>.

Appendix C: Setting up a new Macbook

Upgrade your OS

If your computer is not using the latest OS, you should upgrade to the latest OS first.

Installing Python development environment

The best way to manage Python installations these days is a "conda env". This will allow you to manage different Python "environments", where each environment is a set of libraries that you have installed. For example, you can have one environment that uses Python 2.7 and has certain library versions installed, and another environment that uses Python 3.5 and has other libraries installed. Another advantage of conda environments is that you can apply the same procedure on NERSC and other computing centers that support conda.

How to do this:

- Follow the online instructions on installing a conda environment and see modifications below:
    - ***http://conda.pydata.org/docs/using/index.html***
    - (probably) prefer to install the "miniconda" version rather than anaconda
    - Install "miniconda 3". We work in Python 3.8+ these days.
    - When creating environments, use a command like this (note that this also installs recommended libraries):

      ```
      conda create --name py3 python=3 numpy
      matplotlib seaborn plotly pandas flask
      pymongo scipy sympy scikit-learn jupyter
      ```

    - If you want a reference guide to conda commands, try: ***http://conda.pydata.org/docs/using/cheatsheet.html***

Install high-throughput computation environment

Our group has a set of base codebases used for performing high-throughput calculations. Note that if your project does not involve high-throughput calculation, you may need only one or two of these libraries installed – contact Anubhav if you are unsure.

- Install the following packages using a combination of `git clone >>REPO_NAME<<` and `python setup.py develop`. Start with:
    - `git clone https://www.github.com/materialsproject/fireworks`

        - You might need to generate an ssh key for the git clone command to work:
            - `ssh-keygen -t rsa -b 4096`
            - no password is probably OK unless you are security conscious
            - add your SSH key to your Github profile
    - Then:

      ```
      cd fireworks
      while read requirement; do conda install
      --yes $requirement; done < requirements.txt
      && python setup.py develop
      python setup.py develop
      ```

        - note that the middle line is not technically necessary, but will install requirements with anaconda which might be better than pip
    - Repeat the process above but replace “fireworks” with:
        - pymatgen
            - this one might be tricky, because you need a modern C compiler. You can try installing gcc-8 with homebrew, then setting the following environment variables:
                - export CC=/usr/local/bin/gcc-8
                - export LDFLAGS="-L/usr/local/opt/lapack/lib"

        - export CPPFLAGS="-I/usr/local/opt/lapack/include"
    - pymatgen-db
    - custodian
    - atomate (note: this is on the hackingmaterials github site)
    - matminer (note: this is on the hackingmaterials github site)
- If you want, you can automatically source activate your environment in your .bash_profile file. This will automatically load your environment when you open a Terminal. Otherwise, you will start off in your default Mac Python and will likely cause you a lot of confusion

Configure Pycharm IDE

An IDE allows you to be a much more productive coder. It is like a text editor but contains many useful keyboard shortcuts, code-completion tools, refactoring tools, and debugging/profiling tools to help you be more productive. It can also automatically reformat your code to trim line lengths and add proper whitespace to be in-line with recommended Python formatting guidelines.

Pycharm is the group's recommended IDE for Python and they offer a free community edition (CE) that contains all the features you need. There are other programs you might consider like Sublime Text, although those are more like advanced text editors than proper IDEs. Note that there are some advanced programmers that know their way around an IDE but still prefer an editor like vi or emacs with

appropriate plugins. This is fine so long as (i) you are an advanced programmer and (ii) you have first tried an IDE for a few months and really tried to make use of it, but find that it hampers your productivity. Note that most people that think they fall into category (i) do not and I find them making mistakes that could easily be avoided with an IDE. Thus, it is recommended that essentially everyone in the group use an IDE.

After downloading and installing Pycharm and launching it for the first time, you'll be asked some options. I suggest using the default Mac OS X keymap unless you are already very familiar with Emacs or other keymaps. Note that these things can be changed later if desired.

Next, and assuming you've already cloned the source code of your desired repos from Github, you do the following:

1. Create a new project (give it any name)
2. In the menu bar, click Pycharm CE -> Preferences -> Project -> Project interpreter. Change this to your conda Python interpreter. You can also set your default interpreter from File->Default Settings->Project Interpreter and set that to your conda environment interpreter. Do not skip these steps!
3. Click File->Open and navigate to the root folder of one your desired repos (I suggest pymatgen to start) and click open.
    - Make sure to select "open in current window" AND check "add to currently opened projects"
4. Repeat step #3 for all your desired codebases. When finished, you should see in the sidebar all the various codebases.

If you want to try adding some of your own scratch code, then:

- Navigate in the sidebar to your main project folder (folder with your chosen project name). Right-click and click New->Python package. Give it a name.
- The previous command created a new folder. Navigate inside that folder, right-click, and choose New->Python file. When finished, that folder should contain two .py files - __init__.py (created automatically for the new Python package) and your chosen filename.
- Finally, type some code in your new file. It can be simple as `print("hello world")`. To execute the code, you can use Ctrl+Shift+R with default Mac keybindings or go through the "Run" menu for more options.

Now you are all set! There are many things you might want to do:

- Configure the way your Pycharm window looks. For large monitors, Anubhav likes Project Navigation and Version Control at left, main code in center, and "Structure" panel on right which is basically like an outline of a particular Python file showing all the classes, functions, etc. at a glance.
- Explore the various options and capabilities of PyCharm. Appendix D of the handbook has some tips and you can enable PyCharm's "tip of the day" which will really step you through some of the great features.

Other things to do

- Set up your Time Machine backup (make sure you have purchased or received an external hard disk). Just plug your backup drive into your monitor so when you connect to your monitor, you also back up. For detailed instructions, see:

https://support.apple.com/en-us/HT204412. If there are (for some reason) errors in backing up, fix that issue immediately. The lab also has an online backup program which is another safeguard against lost data: ***https://bit.ly/2PVPHfM***.There are zero excuses for not doing this.

- Install MongoDB, if you are using databases.
- Purchase Microsoft office from LBNL software distribution. Anubhav uses Office 2011 since he finds it stable and usable, but others in the group have had good luck with the most recent versions. Ask around if unsure.

Contributing code to software libraries

If you are unfamiliar with how to contribute back to the various software libraries we develop through Github, refer to the procedure described here:

http://bit.ly/2AYeT0j

Appendix D: Some notes being productive with a Mac from Anubhav

Basic setup

- Macbook Pro 13" laptop
- Thunderbolt Display (now discontinued)
- Apple keyboard

- Apple Trackpad - less precise than mouse, but can be very productive if you learn all the gestures (e.g., for web browsing back/forward, for mission control, for swiping between different Mac "Spaces")

I use an Apple keyboard and Trackpad so that typing/navigating is similar whether I am at my workstation or whether I am on my laptop.

Early on, I turned up my Trackpad speed all the way to the max. This means I can very quickly move the cursor all the way across the screen. It took a few days to get used to this very sensitive setting but now I don't even notice it (when other people use my trackpad, they usually freak out…)

There are many options I set to make OS/X more oriented for power users. For example, my Finder window shows directory paths at the bottom, I have sidebar shortcuts to many important locations, I display hidden files, I have a shortcut to copy the path of the current Finder location to the clipboard, etc. There are many settings like these for various built-in OS/X apps, but unfortunately I don't remember them all. Getting a good Finder setup is probably the most important.

Apps I use for programming

- I use the PyCharm IDE. Things I like about PyCharm include:
    - underlining errors
    - underlining code "lint", e.g., spacings that do not follow PEP

- code highlighting / editor features (e.g., when you open a CSS file, lines of code that define a color automatically display a swatch preview of that color)
- a nice and powerful search tool (regexes, find/replace in certain files, easily filter through results visually and categorize by what type of file they occur in)
- autocomplete (ctrl+space)
- autofix errors, i.e. red underline stuff (option+enter)
- follow definitions of variables, methods, classes (Cmd+b)
- quickly open classes (Cmd+o) and files (Cmd+shift+o) and variables (Cmd+option+o). Or simply tap shift twice to search across everything.
- go back to previous/next file being edited like forward/back on a web browser (Cmd+[or Cmd+])
- the "find usages" command
- quick documentation lookup (F1)
- fixing all the various spacing / formatting issues automatically (Code -> Reformat code and Code --> Auto-indent lines).
- code refactoring
- structure view of code. I usually have "Project" view and "Version Control" at left of window, code in middle of window, "Structure" view at right of window, and "Todo", "Terminal", and "Python console" at bottom of window (along with search results).
- debugger (sometimes, typically only for heavier debugging issues)
- easy IPython console to test code snippets

    - there are other commands that I use, but those are the ones I use most often
    - note that others use the git integration, which avoids needing to leave PyCharm to pull/push/etc. to Github, but I prefer Gitbox for this purpose. I just leave Gitbox in an adjacent "Space" (virtual screen) on my Mac and use a swipe gesture to quickly flip between screens.
- MongoHub (for visually exploring Mongo databases). Other options are Robo 3T (***https://robomongo.org***), NoSQLBooster (free version), Studio 3T, and MongoDB Compass.
- Gitbox (I almost never use the Git command line; Gitbox is unique in that it is really easy to preview changes to the remote before pulling them in. It is also the most intuitive Git tool I know of). Another one to consider is GitKraken which has a free version and looks fun (if I were starting from scratch, I might learn GitKraken).
- Patterns (for tricky regexes) - although the free web site ***https://regexr.com*** is just as good.
- Smart JSON editor
- Balsamiq Mockups - wireframes
- TextMate as a regular text editor.

Apps I use for Science

- CrystalMaker (and sometimes Vesta) - crystal structure visualization
- Zotero - reference management
- MS Office Suite

Note that I use Zotero not only for reference management but also for taking notes on articles. Some of the things I like about this system:

- The notes are kept together with the articles, so I can quickly bring up the article if I am reviewing the notes
- I can easily see which articles I took notes on because I can add a column header in Zotero to show the number of notes for each article.
- The notes are very quickly readable as plain text (versus hunting for notes on the PDF itself) and I can export them easily via BibTeX export. This retains all the notes I took in the BibTeX in case I need to migrate to another system later.

This system isn't perfect but has worked well enough so far.

Apps I use for working more quickly

- Alfred - application launcher, quick file opening, quickly go to a web site. Note that if you don't use Alfred, the built-in Mac Spotlight now includes some of its features.
- Trickster - for easily calling up recent files, e.g. drag a recent file from Trickster into an email
- Default Folder X - the most useful feature of this is that it can add a sidebar to your save dialog that lets you access recent folders. This is 95% of the time where I want to save something.
- Fantastical - for quickly scheduling meetings or looking at my schedule

Apps I use to keep things organized

- Evernote

- 2Do - allows for complex todo lists, but also easy to use and intuitive. All my tasks are managed here.
- Screenshot Plus - Mac widget for quickly capturing screenshots (if like me you can't remember the keyboard shortcuts)

Misc Apps I use

- Bartender - allows you to clean up and reorganize your (top) menu bar; especially useful on a 13" screen. (Also check out a free competitor called Vanilla)
- Moom - keyboard shortcuts for half-screen, full-screen, etc. like Windows has had since Win7. Magnet is an alternative that works well, though it is paid.
- Mousepose and IMovie - screencasts
- Tomato One - if I find it hard to be productive or am avoiding doing something, I revert to Pomodoro method with 40 minute sessions and 10 minute breaks
- Focus - for sometimes restricting internet browsing if I really can't focus (usually combined with Tomato One)
- Safari for web browsing (I find the experience to be very visually smooth and pleasing, e.g., when paired with Trackpad Gestures. For example, a two-finger pinch shows all tabs in a window.)
- Pocket - for saving web pages to read later, and then usually never getting around to it
- Time Machine - not only for backups, but also for sometimes recovering past versions of files that might have gotten accidentally changed / overwritten.
- CrashPlan - online backup (also consider BackBlaze)

- Inbox When Ready - a Chrome extension that helps control the flow of your email (requires checking your GMail via Chrome)
- Spotify - music
- Pixelmator - image editing
- Not an "app", but I subscribe to "10 things you need to know today" e-mail newsletter by TheWeek which contains enough news that I don't need to check it throughout the day.

Appendix E: Our open source software philosophy

"If you want to go fast, go alone. If you want to go far, go together".
- Attributed to an African proverb

Although we develop both open and closed source pieces of code in our group, we try our best to release any software that is potentially useful to more than one person as open source. This ends up being almost all the software that we write except perhaps code written to conduct a specific scientific analysis.

Benefits of open-source software include:

- authors can include the code in their portfolio for future job applications
- you get recognition from the community of users of your code as well as personal pride
- more users means more bug reports - this sounds scary but is in fact very useful and important for your own research

- outside developers can contribute fixes and features, so your code gets better for free
- much less friction - easy to share code, fork it, etc. without needing to set up permissions or access. Easy to distribute and install the code, e.g. via PyPI
- many services like CircleCI and PyCharm offer their products for free when the codebase is open source. Not only does this save money, it more importantly saves a lot of time in coordinating purchasing requests through LBNL that need to be renewed.
- your own programming will automatically improve because your code is open source and public. You will be more likely to write documentation and write clean code if you know it is for the world and not only for yourself. This will also encourage writing the code in a more general manner rather than specific to your application.
- you can write a paper about your code whenever ready. There is no separate process of "making the code open source" if it is already open source from day 1.
- it is the right thing to do for the betterment of the research community!

Clarifying common misconceptions about open source code:

- Writing open source code almost never exposes you to getting scooped or having some outsider leapfrog you in research. First of all, it is very rare that an outsider will use your code rather than make their own, especially if you do not advertise your code. Most of the time, you will have the opposite problem - i.e., to convince people to use and trust your code. Second, as the code author you are the expert in the code. Even when there

is an outside user, it is rare that they are as proficient as you in the use of the code. Third, the majority of people are friendly and not as manipulative as you may think.

- Open source code doesn't need to be perfect, nor does it even need to be any good. Often people think that they will make a code open source when it is "ready". This is not the right approach; code does not need to be "ready" to be open source.
- Publishing a code as open source doesn't mean that you need to support the code or vouch for its correctness. You are offering the code publicly without any guarantees whatsoever, and you don't have any additional obligations to anyone. However, if you actively want your code to be used by the community and extended, then be prepared to document and support your code, and to help users and resolve their problems. But this is a separate decision. It is perfectly OK to have an open-source code for which you provide no support so long as you don't try to advertise it for more than it is. It is still better than having that same code be closed off.

Appendix F: 10 of the most common Python mistakes I see from scientist-programmers

There are many, many books and articles on writing better Python code. Please use those if you want to really desire to become a good programmer. Here, I am just focusing on some of the most basic things

that I think are particularly relevant to the types of scientific programmers we get in the HackingMaterials group.

Note: I used this site: ***http://markup.su/highlighter/*** to help write code blocks with coloring. For additional flair, you might also try using: ***http://instaco.de***

1. **Prefer data structures that don't require memorizing array indexes.**

Don't use a data structure (like a list/array) that requires one to remember that "index 8" is the species string and "index 1" is the coordination number.

Bad:

```
my_data = ["Fe2O3", 6, 5, 43, 4.1]
cell_volume = my_data[1] * my_data[2] * my_data[3]
is_insulator = my_data[4] > 3
```

Better:

```
my_data = {"formula": "Fe2O3", "a": 6, "b": 5, "c": 43, "band_gap": 4.1}
cell_volume = my_data["a"] * my_data["b"] * my_data["c"]
is_insulator = my_data["band_gap"] > 3
```

Notice how much easier it is to follow the logic of the code in the second example?

You can also use a pandas *DataFrame* object if you have lots of data and don't want to repeat the same column headers many times or a *namedtuple* if you just want something lightweight.

2. Document all classes and methods in a standard format

It is really important that all classes and methods are documented. Code is much more often read than written (a tenet of Guido van Rossum), so it needs to be readable and understandable. If you don't know what format to use, try the below:

http://bit.ly/2nAxlT0

You should also pay attention to the format already being used by a particular package.

3. Inside of classes/methods, write code that is readable without documentation whenever possible

This is usually achieved by writing descriptive variable names, function names, and good interfaces to functions. As a small example, why do this (requires documentation to tell user what my_d represents):

```
my_d = {"Mg": 3, "Ag": 8, "Li":4}  # dict of el. symbol to coord. number
all_element_symbols = my_d.keys()
all_coordination_numbers = my_d.values()
```

when you can do this (same clarity in first line, better clarity in last two lines, no documentation):

```
elsymbol_coordnum = {"Mg": 3, "Ag": 8, "Li":4}
all_element_symbols = elsymbol_coordnum.keys()
all_coordination_numbers = elsymbol_coordnum.values()
```

Of course, sometimes you will need to write documentation - but usually to explain why, rather than how. Here is the perfect article about that - it is short and sweet:

http://bit.ly/2pgFQXs

Read it!

4. Follow PEP formatting guidelines

Following proper code formatting helps clarify your code. There are a billion PEP rules and you don't have to follow all of them. But at least get the basic ones correct. Like:

- functions/methods are named like this: `my_very_first_method()`
- classes are named by CamelCase like this: `MyVeryFirstClass`
- python files are named like this: `my_very_first_file.py`
- python modules are named like this: `my_very_first_module`

If you use an IDE like PyCharm, it will detect, underline, and automatically fix most of the worst cases for you, so learn to use the feature. There are also tools like PyLint that you can use separately from IDEs (PyCharm basically has a nice wrapper around PyLint).

Also, don't use ugly code separator comments like "`##########################`" or ASCII art - stay clean and professional.

5. Use standard file formats

Use JSON or YAML most of the time if you need a file format, *e.g.*, for a settings file. XML is very heavyweight and quickly being outdated. Don't invent your own strange conventions (like CIF or any other custom file format).

6. Wrap exe code in 'if __name__ == "__main__":'

Python often runs your file even when you don't intend it to, e.g. when loading a module or importing some component of your file. It is important that you don't run code as a "side-effect" of this. Use the `if __name__ == "__main__"` wrapper to prevent this.

7. Be aware of Python gotchas, in particular mutable default arguments

Do you see anything wrong with this?

```
def append_to(element, to=[]):
    to.append(element)
    return to
```

If you don't see it, then you're going to get hit with some strange and difficult to pinpoint bugs downstream in your code.

This is a common Python gotcha (there is lots of discussion online about it)
http://bit.ly/2nijUdp
http://bit.ly/1wfFNKa

8. Write unit tests

Scientific researchers often don't write tests because (i) they don't write large, complex code with many moving parts or many different authors, (ii) they are overconfident about their ability to write correct code, (iii) they feel this will slow them down. Professionals write unit tests because they know that the longer and more complex a codebase becomes, and the more users it has, the more likely that something is going to go wrong down the line and the greater the dividends that are paid from writing unit tests. Unit tests allow code to be automatically tested for bugs every single time anyone makes a commit (continuous integration) and has demonstrated its value many times over in the large production codes that we use and develop - even (and perhaps especially) for ones required to do complex tasks on a deadline.

9. Throw exceptions rather than returning coded results

One of the most common beginner mistakes is to think that their code should never throw Exceptions or Errors. Perhaps this is because in the beginner's mindset, Exceptions are associated with bugs (e.g., they run a code with a bug and see an Exception, so Exceptions are bad). Another issue is that beginners never want their code to interrupt the operation of whomever is running it. So rather than throwing an Exception when their code is given bad inputs, they will return None, -1, or False, so that they don't interrupt whomever is calling their code.

This is bad. If the user gives bad or nonsensical input to a function, an exception needs to be raised and the program needs to stop immediately if the user is not explicitly catching the exception. For example, if you

try to use the Python math library to compute the log of a negative number [`>> math.log(-1)`], it doesn't return None or some nonsense like False or -1. It throws an Error! As a user, the error is much more useful than any other course of option. Think of the alternatives for math.log(-1):

- If the function returned -1, you would have returned an incorrect result (extremely bad). For example if your function computed math.log(x) * 3, and you gave a negative x, your function would return -3 - which looks perfectly reasonable but is completely wrong! This is the worst possible thing you can do.
- If you return None or False to avoid inconveniencing the user, you have just made two mistakes. First, the user doesn't really know that their input was bad; perhaps it is simply the math.log() function has a bug leading to the strange output. The second and more important issue is that a user might want to run the math.log() function over an array of 1 million integers, and then do a lot of complex processing after that. If math.log() didn't immediately throw an error when encountering a negative number in the 1 million integer array, then the code would keep proceeding with nonsensical results and the user might finish 5 or 6 additional processing steps downstream before the code finally chokes and dies because there are strange "Nones" or "Falses" where there should have been numbers.There are even more dangerous situations that can occur, like a second library to compute standard deviations that ignores None values in the array. Then the user has unwittingly taken the standard deviation of only a subset of the data and never even knows that there was a problem in the pipeline. At

that point, it becomes extremely tedious to trace back the source of the error.

Code should fail immediately when there is invalid input. In general, the further the "distance" from the actual place where the problem originated and the point of failure/exception in the program, the more difficult and maddening the debug task. You are doing users a favor by moving program failure right to the place of the problem.

10. Prefer python lists to numpy for simple things

Numpy is great, but it is often overused (leading to worse code). Numpy is great for algorithmic work, for very complex slicing of multidimensional arrays, and for a host of other things, but it is *not as good for creating basic data structures*. Here are some advantages that Python lists have:

- Python lists have cleaner built-in functions and code. There are lot of tools for Python lists, like index slicing and iterations, or functions like sum() and all() that make them very powerful while still very clean. Numpy has even more useful functions and operations than that (e.g., a built-in mean()), and sometimes you might need numpy in order to leverage those features, but there is no need to transform to numpy arrays to (for example) take the sum of an array. Master regular Python lists first before reaching for numpy because you will have much cleaner code.
- Python lists can be easily appended and modified without lots of "filler" like figuring out how long the array needs to be in advance and populating with zeros before modifying values.

This again leads to much cleaner code and is much easier to write and to read.

- Python lists can be easily serialized and deserialized, e.g. to JSON format where they are native.
- Despite the claim that numpy is fast, numpy lists, arrays, etc can actually take a lot of time to initialize - maybe 100X more than default Python. Of course, if you are then going to do heavy processing on that matrix, like diagonalizing a large matrix or doing large matrix multiplications, numpy will absolutely improve your overall performance, perhaps to a large degree. But for simply creating a data structure or taking the sum of a list, you will perform much worse with numpy while writing less readable code.
- Python lists are more universal; they don't require dependencies and they are readable by many more programmers.

Note that this doesn't mean to stop using numpy. Numpy does lots of things that regular Python cannot and it is an extremely powerful and useful library. But for routine file parsing (where being able to append easily is important), data representation (where serialization is important), overall code clarity (always important), and even speed for routine tasks (usually important) the native Python lists often have the advantage.

Appendix G: Giving effective presentations

"I am a successful lecturer in physics for popular audiences. The real entertainment gimmick is the excitement, drama and mystery of the subject matter. People love to learn something, they are 'entertained' enormously by being allowed to understand a little bit of something they never understood before. One must have faith in the subject and people's interest in it."
- Richard Feynman

"I would drop everything to hear him lecture on the municipal drainage system."
- David Mermin, about Feynman

Good presentations have a thoughtful purpose

The first step to giving an effective presentation is to understand their value in achieving goals that are difficult to attain otherwise. Here are just some of the ways in which giving presentations can be be helpful:

- to establish your expertise and to have your name recognized by the people in your field, especially independent to that of your supervisor
- to encourage people to collaborate with you
- to convince people to test your theory/prediction or to influence the research direction of others

- to convince someone or a committee to target you for a job offer or offer you funding for your idea
- to encourage people to cite your paper
- to encourage people to use or contribute to software that you've developed
- to receive useful feedback on preliminary ideas you may have
- to "test" the talk itself, i.e., gauge audience reaction and points of confusion (based on the after-talk questions) to improve subsequent presentations (like a stand-up comedian at a small venue)
- to simply help expand your audience's knowledge about a particular subject and "tell them something you've learned"
- to solidify your own thoughts about a topic!

Before you begin designing your presentation, you should be clear about one or more very well-defined goals you want to achieve by giving the presentation. For example, if you want to encourage people to use software you've developed, you'll need to include slides explaining its capabilities and benefits to the community and as well as how to obtain and perhaps use the software. If you want people to test your theory, you should include slides suggesting how and why people might attempt this. If you want feedback on your ideas, you should further emphasize points of confusion / unresolved problems.

Note that you should not underestimate the value of giving presentations simply to establish your expertise and to promote your work. You may think that doing good research or writing a paper is enough. Unfortunately, this is usually not the case. Take the example of musicians: they cannot simply record an album and sit back expecting a devoted following of fans. They must instead earn their fans by going on

tour and generating excitement about their music, perhaps starting out as a small and relatively unknown "opening act"; if they do a good job, the live act will encourage people to investigate the recordings. It is not much different for science; when you obtain a valuable result, you should go "on tour" and focus on disseminating it.

In summary, first decide on your goals for giving a presentation, then design your presentation around those goals.

Keep in mind context, audience, and length

When deciding on the goal of your presentation, keep in mind context, audience and length.

In almost all cases, you should aim for a simpler presentation that just highlights the main concepts, but refers people to a paper or other document for more details. Meaning, the purpose of the presentation is to get people interested, not to tell them everything. This is especially the case if:

- your presentation is short (e.g., 15 mins). Some of the worst presentations are those that try to accomplish too much in 15 minutes and end up doing nothing
- your presentation is to a general / broad audience. Most people understand that they should simplify for a general audience.
- your presentation is in a symposium with many other presentations. If you are giving an independent invited talk somewhere, you can assume that mostly people will be "present" for your talk. But if you are giving a talk at a conference where there were many talks before yours and many talks afterward,

people's brains are at 50% capacity due to information overload. Keeping your message simpler and using broad strokes really helps make your presentation stand out - with details saved for Q&A or by referring to a paper.

A longer, more detailed presentation is more appropriate when you have a long (e.g., 45 minute) time slot and when you are not in the middle of many other presentations (i.e., people's brains are not already overworked).

Three good presentations

There is no single good presentation style. A good presenter doesn't have to be authoritative or have a low voice. You don't have to change your natural personality to give a good presentation. Here are some examples of people with different personalities nevertheless giving effective presentations.

- Donald Sadoway (15 mins): formal, authoritative, high salesmanship yet also unconventional/creative with well-rehearsed spontaneity (e.g., use of blackboard): ***http://bit.ly/1WuEkeK***
- Walter Alvarez (22 mins): approachable, casual / unpolished yet poetic, with several tangents - yet inspiring wonder in the subject: ***http://bit.ly/2ov568E***
- Mick Mountz (12 mins): not necessarily a "natural speaker", but makes a boring subject (packing boxes) fascinating through a great presentation structure and slides: ***http://bit.ly/2oSwO1y***

None of the talks are perfect, and thinking about why will help your own presentation skills. However, the above talks are able to get the audience interested in the problem and invites them to briefly join them in their field of study. This is in contrast to talks that try to oversimplify concepts or try to sugar-coat them with fancy graphics - i.e., present problems solutions in a way that is different than the way they themselves think about it. This is a mistake that many other TED-style talks or cable TV documentaries about science make. These talks also include small tangents that could easily be the subject of other talks. You can learn a lot from seeking out and taking notes on good presentations.

Two presentations "close to home"

Here are two of my previous presentations:

- Alvarez symposium (2011, immediately following my PhD) - this was a general audience (non materials scientists) and the purpose was to honor the late physicist Luis Alvarez. I'm clearly a bit nervous but have pretty good slides to lean on. This is also the first time a talk of mine was recorded and I learned a lot by watching it. ***http://bit.ly/2pWKiuW***
- University of Wisconsin summer school (2014, ~3 years past my PhD)- the audience were all materials scientists interested in learning modeling techniques. By now I am more comfortable in giving a talk and have corrected some of the errors of the previous talk. ***http://bit.ly/2pEflZm***

Presentation checklist

Here is a checklist you can use to improve and verify various aspects of your presentation.

Easy things to do:

- ❑ **Number your slides.** Numbered slides make it easy to refer to specific slides during the Q&A or feedback period.
- ❑ **Confirm all font sizes are large enough so that even people in the back of the room can read them.** One good way to do this is to make the fonts way too big, then reduce the size until manageable (rather than starting too small and increasing from there, which in 90% cases leads to fonts that are still too small).
- ❑ **Write slide headings as snippets that contain useful information.** A bad heading would be "*Effect of +U parameter*". A good heading would be "*Band gap and VBM d-character increase with +U parameter*". There is a style of slide called "evidence-assertion" that is generally very effective and should be used often.
- ❑ **Sharpen your images.** Rescaling images in Powerpoint tends to make them appear soft, and projectors can also be less sharp than your display. Use "Format Picture->Corrections->Sharpen" to sharpen all your images and make them clearer for display.
- ❑ **Make sure you are not showing too many plots.** The audience is seeing your plots for the first time. Any new plot you bring up will take at least a few minutes for the audience to digest. You need to explain the axes and give them time to understand the scale, you need to explain the various curves being plotted and how they relate to one another, and finally you need to explain the overall message of the plot. Because each plot should be a major chunk of time, you don't have room for too many of them in most talks. One big mistake I see is that people want to keep their presentation to "10 slides", but then they have 4 plots per slide.

Intermediate things to do:

- ❑ **Minimize the use of written text.** Research demonstrates that your audience cannot read text on your slide and process what you are saying at the same time. Every second they are reading, they are not listening to you. In contrast, audiences have no trouble simultaneously listening and processing visual information (diagrams, images, etc.). Design your slides to account for this quirk.
- ❑ **Convey information through multiple "channels".** Ensure that critical information is not only contained in your speech/delivery but also through a visual channel (images or short text phrases / conclusions). People may not be able to hear you or might be distracted by their own thoughts for many moments in your presentation. Or, they might not understand a visual diagram and be helped by reiterating the point a different way through your dialogue. Having multiple channels maximizes the chance that they will receive the signal of your talk even when there is external "noise". More advanced presenters will use body language or position as another "channel" through which to convey information.
- ❑ **Rehearse your talk for "flow", "momentum", and "energy" and cut slides that disrupt flow.** Rehearse your talk, paying attention to the slides/sections in the talk where you are (i) struggling to explain a slide, (ii) where your energy / enthusiasm level drops, or (iii) the momentum of the talk seems to be slowing down. One symptom of such struggling is talking quickly in order to explain everything on the slide. 90% of the time, I find that *removing* such slides from the talk (i.e., moving it to an Appendix/supporting slides) is the best course of action

- even if I initially think that slide is important. Rehearsing the section again usually reveals you can maintain the energy and flow of your talk much better without the obstruction of having to explain that slide, and you can explain away the missing concept in a sentence or two while retaining the momentum of the previous slides. If it turns out the slide was in fact critical, then perhaps re-design the difficult slide as multiple slides to more gradually set up the concept.

- ❑ **Memorize the order of your slides; use "Slide sorter view" to help.** During every point in delivering your presentation, you should be able to picture what the next slide in the presentation is. If you can do this, you are more likely to speak in a way that naturally connects between slides rather than abruptly stops/starts between slides. Some presenters use "presenter view" in Powerpoint during their talk to help with this, but I would say that depending on this feature is less likely to lead to smooth explanations than memorization. To mentally remember the order of slides, I stare at the slide deck in "Slide Sorter" view. The "Slide Sorter" view can usually show me most or all of the presentation at once since each slide is a small thumbnail, and I can easily see the visual overview of essentially the entire presentation. Thus, I can remember the visual arrangement of slides by studying the Slide Sorter view and can roughly flip through the presentation in my head.

Advanced things to do:

- ❑ **Video record yourself rehearsing the talk and watch yourself.** Although you may find this uncomfortable or strange, you will learn much about your presentation style and areas to

work on. Often, Anubhav will do this during your practice talk with him.

- ❏ **Connect your talk with the other talks in the session.** You should modify your pre-rehearsed presentation based on the other talks in the session. You can move quickly through topics or introduction that have been covered a few times before and you can highlight how the other talks connect with your work.
- ❏ **Use an app like "Color Oracle" to see what key figures in your slides look like to color blind audience members.** About 8% of the males in your audience likely have deuteranopia and having figures that account for this can be useful. There are color palettes for color blindness that one can find online, e.g., ***http://colorbrewer2.org***. You likely won't want to do this for all your slides, but might be worth it for showing key results.
- ❏ **Use an app like "Orai" to improve your vocal delivery.** Orai will have you read sample text and analyze your speech for clarity, pacing, and emotion.

Miscellaneous advice

- Make sure you clearly "sell the problem" before presenting your work and your solution. In many good talks, the problem is described in a way that the audience feels (i) that the problem is important and (ii) that they understand the problem well enough to start brainstorming their own solutions before you present your solution (something like a mystery novel). Note that this is different than a paper/written document that simply tries to convey information efficiently. In a presentation, it is

useful to have some drama and create some audience engagement.

- Some techniques to help give better presentations are:
    - add comparison points (e.g., 1 kilogram of this material can store enough energy to power a light bulb for 10 hours; or, the amount of computer time involved, if done on your laptop, would take 50 years)
    - use analogies, particularly when presenting to a non-specialist audience (i.e., the material soaks up lithium ions like a sponge)

Appendix H: Writing effective papers

Here are some miscellaneous tips for writing papers. There also some good tips in the short article below (which echo many of the things I write below, such as the importance of a discussion section and how it is better to be short than long):

https://serialmentor.com/blog/2013/8/29/writing-a-scientific-paper-in-four-easy-steps

Writing style

Active vs. passive voice

This is one of the unending arguments about writing style for scientific papers (active: "We conducted a DFT study..." vs. passive: "A DFT study was conducted ..."). In general, I prefer the active voice about 80% of the

time, with the exception of the Methods section for which I typically use passive voice. This is in-line with most modern recommendations from multiple sources, although one can certainly find disagreements. Note that journals like Nature state that they *always* prefer the active voice.

Science is quantitative - give numbers

Always provide numbers, not just text.

Bad: "The computations and experiments agree very well."
Good: "The computations and experiments agree very well, with an r^2 of 0.89 and a mean absolute error of 0.2 eV (Table 3)."

Bad: "The Seebeck coefficient for layered chalcogenides was previously found to be very high,[3-5] making these systems interesting for thermoelectric applications."
Good: "The Seebeck coefficient for layered chalcogenides was previously found to be very high (in the range of 300-400 microvolts/K),[3-5] making these systems interesting for thermoelectric applications."

In particular, the abstract of the manuscript should contain all Very Important Numbers (number of materials investigated, major result numbers, number of good compositions found, etc.) as well as important compounds, techniques, etc.

Use specific verbs

One of the ways to tighten and polish a manuscript is to use more specific verbs.

For example, instead of "We *study*", try one of the following:

- apply, assess, calculate, compare, compute, derive, design, determine, develop, evaluate, explore, implement, investigate, measure, model

Instead of "Figure 1 *shows*", try one of the following:

- plots, illustrates, presents, exhibits, demonstrates, indicates, reveals, depicts

Typically, such refinements would come at a later stage of the manuscript and is not something to worry about in the first draft - although if you write often enough, you'll gain the ability to use more specific verbs in earlier drafts.

Methods section

DFT based

If you are writing a DFT-based methods section, be sure to include:

- DFT code used and version (e.g., VASP v5.2.x)
- version of software(s) used to do other things (e.g., atomate v0.9.1)
- pseudopotential type (preferably with more details if multiple variants with different # of core electrons exist)
- functional(s)
- major functional parameters (including +U values or HSE mixing parameters)
- k-point mesh (or algorithm used to determine this)
- electronic and ionic cutoff energies (or algorithm used to determine this)

It's better if you can list all this explicitly in the text (either main or supplement) rather than refer to some other paper that has details. Not everyone has access to all our papers, plus it is just a pain for someone to track down information across many papers.

Machine learning based

For machine learning papers, be sure to include:

- size and details of full data set
- data cleaning, filtering, normalization, etc.
- strategy to do train-test split or train-validation-test split, CV, nested CV, etc.
- features/descriptors tested
- any feature normalization, feature processing, or feature selection strategy employed
- algorithm(s) chosen (e.g. random forest)
- hyperparameter selection strategy, and values of optimal hyperparameters if possible to summarize
- ideally, a link to a code repo where someone can run the same trained model on their own data set

Discussion section

"Meaning is the relationship of the foreground figure to the background"

- Bruce Lee

I've found that many researchers, even senior ones, either skip a Discussion section or do a poor job of writing one. Note that it is certainly possible and sometimes advantageous to folding the discussion into the results, but in the majority of cases I've found that this strategy

is advocated by those who don't write good discussion sections. Thus, I suggest first writing a separate Discussion section to ensure that the discussion is strong on its own, then folding components into the Results section later as needed.

Some of the things to do in the discussion section:

- Put your work in the context of past results in a way that is deeper than the introduction (i.e., now that you have presented your results, you can really show how those fit in or don't fit in with prior results). Do your results match, modify, or disagree with prior results?
- Explain any limitations of your work as well as whether those limitations could potentially change any conclusions or limit the range of applicability of your work.
- Show and explain which of your results can be explained by existing theory / chemical principles / paradigms.
- Highlight which of your results can NOT be explained by existing knowledge in the field. Provide your own thoughts on any outlier points / unexplainable results. These do not have to be correct, just plausible. If such thoughts are testable by further computations, you should test them and show the results (even negative results can be shown to rule out possibilities). As stated by Richard Feynman: "*The exceptions to any rule are most interesting in themselves, for they show us that the old rule is wrong. And it is most exciting then, to find out what the right rule, if any, is.*"
- Offer any new design rules you can come up with and discuss any important tradeoffs that might need to be made.
- If you offer any computationally testable hypotheses, then test them, don't just theorize. For example, let's say you are trying to explain why compound A has a larger band gap than compound

B. You hypothesize that it's because compound A has a smaller cell volume than compound B. Many people will just leave it as that. But what you should do is to actually compute both compound A and compound B at the same cell volume (either at that of A, that of B, an intermediate volume, or all 3) and show that the discrepancy between band gaps goes away. Then you can chalk up the discrepancy to volume confidently instead of just theorizing.

You can see an example of a discussion in one of my earlier papers: "*Relating voltage and thermal safety in Li-ion battery cathodes: a high-throughput computational study*".

Conclusion section

Many researchers copy-paste and re-word the abstract for the conclusion (or vice-versa). However, this section can include more. While you should certainly summarize the paper's main results, don't be afraid to also use this section to speculate about the future. This includes:

- how your results might be applied to various materials classes or analyses
- suggestions for future study, either to be conducted by yourself or by other researchers
- what kinds of further advancements would be most useful or needed

Acknowledgements section

It may not seem like it, but the acknowledgements section is one of the most important parts of the paper when it comes to research funding. In fact, funding agencies usually ask us to write out the *full acknowledgements section* for every paper we published with that funding. This text decides the degree to which we get "credit" for the publication in the eyes of the funding agency.

For the acknowledgements section:

- Thank anyone for help with the project that is not already an author on the paper - e.g., if someone helped introduce you to a topic or method, or contributed in some small way
- Thank any software library that you did not already cite / mention in the main paper
- Ask and confirm with Anubhav the text that should be used to acknowledge the funding agency. If this is primarily our work, we should have some text that says the project was "intellectually led by" or "primarily funded by" our funding source. There is a big difference in the way we report papers funded with those words in the acknowledgements versus missing those words.
- If you used any supercomputing resource, find the guidelines for acknowledgement and make sure to follow them. This text is often needed in order for the paper to qualify for computing time renewal grants. For example, the text for NERSC is listed here: ***https://bit.ly/2Q5eAVV***

- Some guidelines on funding acknowledgements from LBNL's MSD, which is also applicable to us for the most part, can be found here: ***https://bit.ly/2CMhhHU***

Paper checklist

Here is a checklist you can use prior to having a "final" version of your paper.

Pre-"final" checks:

- ❑ **All the numbers in the manuscript are correct.** Most researchers double-check their text and wording multiple times before paper submission, but don't specifically and separately check all the numbers in their text. I've identified many errors in "final" manuscripts simply by having a separate check for the numbers without paying attention to the text.
- ❑ **All acronyms are defined during their first use.**
- ❑ **All important prior works and research groups are cited.** You should give credit to prior works where it's due and give readers a broad perspective of the field and various approaches. Also note that this can often be the difference between a referee supporting your work and rejecting it.
- ❑ **The abstract and conclusion summarize all the Very Important Numbers.** For example, these sections summarize how many materials were studied, the value of any outstanding property measurements, or the number of candidates recommended for further study.
- ❑ **The figures/tables and captions alone tell the story of the paper.** A reader should understand what your paper is about and its major conclusions from the figure and figure captions

alone. If you want an example of how to do this well, look at any National Geographic feature article. The articles are usually lengthy, but one gets a good feeling of the article just by looking at the pictures and captions.

- ❏ **The acknowledgements are complete.** This includes funding sources (very important for the PI), computing resources (very important for getting computer time), people that helped, and any software you found helpful. See the specific section on proper acknowledgements!

Miscellaneous advice

- Don't feel like you need to start with a blank page and start typing out a manuscript, start to finish. This rarely ends well. Instead, begin by outlining sections, then gradually filling in details and polishing things over time. You can (and probably should) send Anubhav an initial outline (including planned sections, figures, and main points) prior to do any serious writing. This also works different areas of your brain and helps prevent "writer's block", which is usually caused by being fearful that any step you take won't be "good enough". Outlining and drafting helps remove this block. The best visual demonstration to see this is to view how even an accomplished artist creates a final painting: ***https://youtu.be/Lye7kK8iOR8*** . The final product is stunning, but the initial stages are unimpressive - just outlining, rough shading, etc. Thus, the key to a ***good*** final product is to (i) start by outlining and (ii) keep polishing, keep improving, keep iterating. The key to a ***great*** final product is simply to continue thinking about, polishing, and improving the manuscript even after the product looks very good. For

example, in the video linked to above, the artist already had a very good painting at the 5:00 mark of the video (about 80% through) and could easily have stopped at that point. Yet the artist continues improving anyway to end up with something even better.

- For sections of the paper that are meant to be conversational (i.e., the Introduction, Conclusion, and Discussion), consider speaking your written text out loud. It will help you identify portions of the writing that are best re-worded for clarity.
- Keep it short and interesting, especially for early drafts. You want to maintain a high level of reader interest and attention throughout the paper - keep them excited! One can always add more information that really feels missing in a later stage of the draft. If there are tangents or very boring but necessary things to include, they can go in either the methods or the supplementary information where anyone can find it (usually even those without a journal subscription!) and where they won't interfere with the narrative of the main results. The main results should be exciting!
- Try to tell the whole story in two ways. First, a reader reading the main text (but never looking at figures or captions) should understand the whole story, just not all the details. This means that your main text should contain guided tours of your Figures (almost like an audio description) such that even someone who is too lazy to look at the figure can keep reading and know what was presented in the figure. Second, the figures and figure captions should operate independently of the main text. A reader just skimming your figure and figure captions without reading any of the main text should get a sense for all the

important results, and the captions should give enough information to let the reader know what they are looking at.

Tips/checklist for sending papers to Anubhav for review

Here are some things you should to do to ensure a timely review of your paper:

- ❑ **Include page numbers in the manuscript.**
- ❑ **Ensure that what you are sending for review is as short as possible.** For the first review, perhaps this means just an outline with Figures and Figure captions rather than a long paper with a lot of text (that may need to be changed, reorganized, etc).
- ❑ **Ensure that the paper is as short as possible.** Ideally, this means an abstract of ~250 words (no more than 300). For the main text, aim for 4500 words and no more than 5500 words (including figure/table captions). If you have more, start putting things in the supplement. We can move things out of supplement later if really needed. If you have trouble deciding what to move to the supplement, talk to Anubhav about how to go about it. *Please don't send Anubhav papers that are >5500 words long for review unless previously agreed this length of paper was necessary* - it means you haven't yet done your job of reducing the paper down to its essence.

Appendix I: Mechanics of writing papers in Microsoft Word

Many researchers gravitate to LaTeX because of poor experiences with Microsoft Word. With few exceptions, I've found the poor experiences to be due to either (i) not using a good template (journal templates are notoriously bad), (ii) lack of knowledge with using MS Word properly - e.g. to reference figure captions or add citations, or (iii) using a very outdated version of Word (i.e., prior to Word 2011).

That stated, there are certainly good reasons to use LaTeX as well. Especially when a document has multiple collaborators (all of whom need to add references), I tend to prefer LaTeX plus Overleaf which is supported by LBL: ***https://www.overleaf.com/org/lbl*** **.** The Overleaf tool keeps getting better making LaTeX very attractive these days.

For now, let's stick with Word unless there's good reason not to. Here's how to write a good-looking, easy-to-manage manuscript in Word. Note that some of the instructions may differ slightly depending on your version of Word.

Start with a visually attractive template

Rather than using journal templates, which are often buggy (and the source of many frustrations with Microsoft Word), it is better to write and modify your manuscript using the group's template. To get started, download the style from

https://hackingmaterials.lbl.gov/stuff/word_styles_v3.zip and start writing in "AJ_paper_v3_example.docx" and save it as your own file (alternatively, you can install the style file or template file included in that zip archive). If the document looks strange, you may be in outline view: simply go to View and click on Print Layout, then adjust the zoom level to your liking (for large monitors, I often prefer two pages on the screen).

Add sections and subsection headings properly

In the "Home" tab, there are various types of formatting styles including Normal, Heading 1, Heading 2. To fully take advantage of this feature make sure that you enter main section headings with Heading 1, your subsections with Heading 2, and all main body text with Normal. In this way, headings will renumber themselves as needed as you add more headings and sub-headings. To reference a particular section, use the "cross-referencing" feature described later in this document (do not just manually type out the section/subsection name or number). Such cross-references will automatically update as headings change.

Insert figures and tables and their captions properly

Use the following procedure to insert and place figures and tables:

1. At the desired point in your manuscript, start a fresh line of text and insert your figure or table using the Insert->Photo or Insert->Table command (or drag/drop a figure from your file system, or copy-paste a table from Excel, etc..)
2. Right-click on the object and then select "Insert Caption" (choose the proper label; e.g. Figure 1 vs Table 1) in the next line

and write your caption. Note that these captions will auto-update and auto-renumber.

3. (optional). If you want to resize or reposition your figure/table and caption, then with your mouse select both the object and the caption and then Insert > Text Box: Now the object and its caption should be nicely fit into a text box which you can simply move to different places in your document using your mouse. The text box is particularly helpful when writing in formatted documents as it can be moved to your desired location. Furthermore, there are formatting options in the text box that allow the text to flow around the text box in your desired style (right-click the entire text box and select from the various "Wrap Text" options).

To refer to a figure or table in the main body of the document so that numbering is handled automatically, see the next section on cross-referencing objects.

Insert equations and equation numbers properly

In earlier versions of Word, it was a pain to add equation numbers. In Word 2016 or greater, it is thankfully very simple:

1. Navigate to the point in the document where you want to insert an equation
2. Insert an equation and write out your equation using the Equation Editor tool
3. *Staying within the equation*, go to the end of the equation and type a space followed by #(1).
4. Hit "enter" to complete the equation

Word will automatically number your equation.

For more, including automatic numbering, see this link (and in particular the response from "Chris C"): ***https://bit.ly/2JccLG5***

Cross-referencing objects: sections, subsections, figures, tables

When referring to a section, subsection, figure, or table in your main body text, it's important that any numbers and/or quoted text are automatically updated so that you don't need to manually change these references (e.g., do not manually type "Figure 1" in the text!!). Instead, to cross-reference items:

1. Go to References > Cross-reference
2. Choose your Reference type first (e.g Figure) and choose "Only label and number" under "Insert reference to:" part and then click on Insert. The cross-references that are created this way should be updated automatically even after you add or remove an object of the same type before the current object. Note that there are multiple styles of cross-reference - e.g., number only, label and number, etc.

Such cross-references will automatically renumber/update as needed. In rare cases, Word fails to automatically renumber/update cross references. The easiest way to force a refresh is to open the "Print Preview" dialog.

Citing articles

We suggest the use of Zotero to organize your research library and to insert citations into Microsoft Word via the Zotero plug-in (or perhaps create a BibTeX library for use with LaTeX). See the Zotero documentation for more details.

Troubleshooting

When you move equations, tables, and figures around the caption numbering should automatically update to reflect their positions in the document. If this does not happen in your cross-references, go to File > Print, then close out of the print menu (without printing). This can fix issues of cross references not updating on-the-fly. This trick may make take a couple tries to work.

Appendix J: Managing the group website

It is currently a single-step process to update the group website:

- update the code in the hackingmaterials.github.io repo
- The docs will automatically deploy to Github pages.

Appendix K: Group library

We have several technical books in the group that you can borrow (just contact Anubhav). The only condition is that you should only borrow books when you intend to read them. It is all too easy to take a book with the *intention* to read it and let it simply take up space on your desk for a year. So if you find yourself not using a book, just return it back.

A list of titles and some brief notes are available on an external site:
https://bit.ly/2Nx8H31

For reference information, the LBNL library also maintains a list of electronic database subscriptions:

https://bit.ly/2HCePDQ

Appendix L: Staying up to date on research and literature searches

The rate of publications keeps increasing every year, and it is now becoming difficult or impossible to keep pace with all the latest developments in any given field. Here are the main tools I use:

- Set up e-mail alerts for new publications from the main players in your field (I do this through Google Scholar, but there are other methods).
- Use article recommendation services to help you find articles you missed from the above alerts from time to time. I prefer

Google Scholar's recommended articles feature - they are usually spot-on for me.

- Reference management softwares like Mendeley and Zotero have their chrome extensions, which can help to download and classify the new papers into the corresponding folders by just one click.

If you are new to a field, one way to try to get caught up is to find a review article and work your way through that. Hopefully, the review will cite some articles that you can follow up on, and those articles will in turn cite other articles that look interesting to follow up on. If not, try a different review article as a seed. You can also do a reverse search, i.e., see what articles have cited a particular article (e.g., use the "Cited by" featured in Google Scholar). Once you've read through a dozen or so articles, you should start to have a pretty decent grasp of the main works and ideas in the field.

There are also more formal resources and databases for doing literature and data searches. I would highly recommend look at the "Materials Science" resources listed by the LBNL library: ***http://bit.ly/2HCePDQ***

Appendix M: some miscellaneous things

Some miscellaneous items are on an external site (***https://bit.ly/2HNtOLQ***) rather than this handbook. They include:

- Some exercises to become familiar with materials informatics, including pushing code to git and using Python data analysis tools.
- Instructions on printing posters
- Instructions on managing the group web site
- Group Figshare

Appendix N: How to do well as a graduate student

A lot of MSE graduate student specific information would be covered by either the MSE department (e.g., the Graduate Student Handbook, which is updated every few months) or the Graduate Student Council (MSE GSC). However, you might want to ask the group members on how they actually managed to do things. If this is the case, feel free to contact graduate students in the group!

Aside from specific Q/A, there are some general guidelines (compiled from Anubhav, alumni, and current students) which can help you do better as a graduate student.

Preliminary Exam and Classwork

Q. Recommendations on what classes to enroll & How do I enroll in classes?
A. In general (but not always), a 1st-year MSE PhD student would want to take MSE classes in their first semester, especially considering that

they would take a preliminary exam in January. MSE 201A is typically what most students take, feel free to take any other courses you want to learn/remind yourself of "broad" knowledge in Materials Science. Taking 1-3 classes in a semester would be general, but it is not very typical to take three or more courses a semester (you would probably want to get familiar with your project). You can also keep in mind that you can add/drop classes until some time after the semester begins. Basic instructions on signing in to classes is provided in below link: https://registrar.berkeley.edu/registration/enrollment

Here are some useful courses if you want to learn:

1. Core tenets of MSE: MSE200A, MSE202, MSE204, MSE201A, MSE223
2. Programming: CS61A and B
3. Machine learning: CS200, CS289, CS288, CS285

It's also *really* helpful to take a course directly related to your research project *as soon as possible.* For example, if you're working on catalysis, it is in your best interest to take the surfaces MSE class as soon as you can (preferably in your first semester).

Q. How to prepare for the preliminary exam at the end of the first semester?

A. The preliminary exam may be one of the most important things you need to deal with after you come here. It can also be a source of enormous stress if you are not prepared. The department will tell you about every rule of the exam, but they will not tell you much about how to *actually prepare for the exam.* Thus, it's important to set aside a considerable amount of time to prepare thoroughly!

Here are some things to do if you want to ace the exam:

1. **(Extremely important)** Ask senior students about the content you should prepare for. Knowing everything about every subfield of MSE is impossible, so you need to narrow your focus. Many senior students have questions from previous exams; you should view these as soon as you can and structure your study loosely around them. For example, one thermodynamics examiner is notorious for examining students exclusively on Ellingham diagrams. If you went into the exam with a broad and impressive MSE knowledge but happened to not know how to read an Ellingham diagram, you would not do well.
2. **(Extremely important)** Organize with your colleagues to do practice (mock) preliminary exams. The exam itself is about MSE knowledge, but also - surprisingly - about keeping your composure while presenting complex information. Practicing in front of others and fielding their questions in real-time is the most effective way of validating your study. This practice will improve your ability to think on your feet and will identify your weaknesses.
3. **(Extremely important)** Know the fundamentals for all the areas flawlessly. You don't need to be up on the latest research to do well on the prelim, but you *do* need to know fundamental equations and ideas *very* thoroughly. For a characterization example, you probably don't need to know every detail of how angle-resolved photoemission spectroscopy instruments work, but you should be able to explain diffraction and Bragg's Law very very well. The best way to do this is to read relevant textbooks and do some of the exercises.

4. Ask faculty examiners which books they recommend. You don't need to read the entire books, but you should at least read (and understand!) the chapters related to the previous questions.
5. During the exam, answer questions simply. Answering with more detail (esp. about topics you are not 100% confident in) is not always better. Providing extra details often encourages examiners to ask much more difficult questions.
6. If you are asked something you don't know, don't say "I don't know" without follow up. Say "I'm not sure, but…" and then give your best guess at an answer based on things you *definitely* know. You can also ask questions to your examiner to clarify the questions.
7. (**Very important**) During the exam, <u>never</u> give up! Occasionally you will get taken into deep water and be totally lost; during these times, it's important to keep your cool and keep answering things to the best of your ability (without making things up or wildly guessing). I have never met someone who had a perfect prelim exam, and I have known many people who had one or two disastrous sections and still passed. Continuing to engage with your examiners (asking for clarification, giving an educated guess, etc.) instead of giving up will drastically increase your chances of passing the exam.

For those who didn't graduate from Material Science, don't worry too much since you can learn everything quickly during the preparation. For those who had a MSE degree, don't be overconfident. It is likely you cannot remember what you learned 2 or 3 years ago in undergraduate study. Again, start the preparation as early as you can. I suggest starting the readings halfway into the semester and starting oral practice 2 or 3 weeks before the exam.

Some students may fail the exam. Don't panic since you have a second chance in summer. I know it is stressful to take the exam again but you can make it. The form of preparation for the second exam is the same as the first one. Just read and practice!

As far as classes go, the survey course **MSE200A** is a great choice for those new to MSE, as it will give you a good overview.

For specific topic areas, the following courses are particularly useful.
Characterization: MSE 204
Materials properties: MSE 223
Bonding and structure: MSE 223, MSE 202
Thermo and phase: MSE 201A
Mechanical: MSE200A (this survey class is often more useful than MSEC211/C212 for the topic matter in the exam)

Qualifying Exam

Berkeley doesn't have a formal doctoral defense, so a lot of the stress and pressure is focused instead on the qualifying exam, where you essentially have to do 3 things:

1. Demonstrate competence in your research field and fundamentals
2. Defend your past research
3. Defend a reasonable proposal for future research

The qualifying exam is probably the most difficult part of the program. Not to worry, though. If you have been making research progress over

your first few years, and if you dedicate some time to preparing your presentation, you have nothing to worry about.

Q. How do I get more information about the qualifying exam?
A. Check the MSE Graduate Student handbook. There are lots of details in there as far as formal procedures go.

Q. When should I start preparing for the qualifying exam?
A. Students usually take the qualifying exam between their 2nd and 3rd years. There is some flexibility in the timing, but the longer you delay it, the more stressful it becomes, as more is expected of you (this is coming from someone who took it at the end of his 5th year!). If you have published a paper with the group, this is probably a good sign you should prepare for the qualifying exam sometime within the next year. Once you have the exam scheduled, you should have *at least* 1.5 months to prepare. Having 2-3 months (or longer) is better (and far less stressful).

More specifically, you should consult with Anubhav (and your UC Berkeley advisor, e.g. Gerd or Kristin) to determine if you are ready!

Q. How do I actually schedule the qualifying exam?
A. You need to first identify your committee members according to the requirements in the MSE Grad Student handbook. You should come up with a concise (~ a short paragraph) proposal for your topic as well. You consult with Anubhav and your UC Berkeley advisor (Gerd, Kristin, etc.) to make sure your topics and examiners make sense. Once you do these things, you can actually schedule the exam.

Scheduling the qualifying exam is sometimes more difficult than the exam itself. You'll need to email your exam committee with at least a few months (if not more) of lead time to figure out when they can all meet during a common time slot. Tools like when2meet and Doodle are very useful for this purpose. It's generally ok to reschedule for good reasons, but try not to! You'll also need to file some formal paperwork according to the MSE GS manual declaring your committee and topic.

Q. How should I prepare my presentation and proposal?
As prerequisites, you should:

- First, be **very** strong on the fundamental science on which your research relies. Know the core equations by heart. The best way to do this is to read textbooks and review article intro sections.
- Have an overview of the current research in the field. It doesn't have to be comprehensive, but it should be decent. Know the core strengths and weaknesses of other groups' research as it relates to yours. Don't be afraid to form your own opinions!
- Have a clear scientific motive (i.e., a research question) for the work you're doing *and* strong justification for the methods you're using to do it. You'll know your research topic is clear if you are able to phrase it as a single sentence.

As far as the presentation and proposal goes, everyone has their own style of presentation. However, there are a few general guidelines that might help keep you on track:

- **Ease your examiners into the presentation.** Your committee will likely be made up of experts from several different fields, and they likely won't know everything about all the topics your presentation covers. Start with some easy slides demonstrating a clear motive for the work you are doing.

Having your examiners be confused is a *bad* thing; they'll ask more difficult questions and be more angry with you. Most questions for your exam will be loaded towards the beginning, so it is better to have the slides be easily defended than overly complex, especially at the start.

- **Keep it simple.** Every word and figure on your slides must be defended. Having less on your slides is better. If you don't know what a plot or equation really means, don't include it. You can always move extra info to the backup slides. If you do want to include more complexity on a particular slide, at least begin with a simple idea and evolve it into a more complex idea as you explain the slide.
- **Propose a clear, succinct scientific objective.** First and foremost, examiners want to know that you know *what you're doing and why.* If you cannot concisely state your objectives, it suggests you don't understand what you're doing. Almost every professor I have talked to says that this is the biggest factor in preventing your qual from going sideways.
- **Emphasize your past work.** If you have published work, you should emphasize it!
- **Demonstrate novelty of your approach with respect to past work.** You should show your work has significant advantages over what others have done in the past. Otherwise, why would you be doing it?
- **Be forthcoming with the limitations of your approach.** All methods (computational or experimental) have limitations. Examiners mostly care not about the limitations themselves (as long as they're not crucial), but rather *that you know the limitations of your approach.* When you explain your slides/proposal, come right out and say how the methods are

limited before they ask. This will demonstrate you are a critical thinker and competent researcher.

- **Schedule at least one practice exam.** You can recruit as many people as you want (Alex has seen practice exams with 30+ people in them), but the most helpful practice is usually more focused (3-6 people). You should also ask your practice examiners to be as critical/argumentative as possible rather than listening to your presentation and asking 1-2 questions at the end. "Overly" critical practice is much more representative of the actual qual.

Q. Is there a particular way I should prepare the written proposal?

A. Aside from the guidelines in the MSE GS manual, not really. Just make sure it presents basically the same information as is in your presentation. You can explicitly include sections like "Background", "Research Question", "Previous Work", and "Proposed Work" for clarity. You can also be a bit more detailed in the written proposal (e.g., cite previous work comprehensively) than in the presentation.

In general, the proposal is not as important as the presentation itself. However, since most of your examiners will read/skim the proposal before the presentation, making a good impression on your examiners with the proposal is a good idea. If your proposal is sloppy, unclear, or fundamentally flawed, your examiners will likely come to your presentation with much more scrutiny.

Q. Is there a particular way I should answer questions during the exam?

A. Yes. Here's some examples.

Examiner: "Have you considered using <Method B> instead of <Method A> you propose here?"

Bad answer: "Uhhm, well, no. My advisor said we should use <Method A>. I remember seeing <Method A> is really good for this kind of problem."

Good answer: "Yes, in fact we have looked into <Method B> but it was limited because of <reasons>. In particular, <reason 1> causes the accuracy to be far off, often more than 200%. A recent study by Author et al. showed <Method B> was both less accurate and more expensive than <Method A> across solids. So, despite the fact <Method A> requires empirical fitting and has problems like <problems>, it seems like the best approach at this time. However, I'm definitely open to suggestions - what do you think about <Method C>?"

Examiner: "I disagree with your conclusions on this slide. <Conclusion A> isn't supportedbecause of <counterevidence>. Can you explain more why you think <Conclusion A> is supported?

Bad answer: "Oh, you're right. I don't know why I put that in here. Sorry…"

Good answer: "I understand <Conclusion A> can be a point of contention because of <reasons>. However, even counting <counterevidence> I believe it is reasonable because of <other reasons>. More, Author2 et al. showed this kind of analysis still yields practical design principles although they are less rigorously derived. I suppose if

we were to be particularly rigorous, we could amend <Conclusion A> to be <Conclusion B>. Then we would have accounted for <counterevidence>. Does this seem more reasonable to you?"

Q. Anything else to watch out for?
A. File your paperwork early. Alex filed his paperwork (literally) 3 minutes before the exam and that was almost a catastrophic mistake. Also, relax! If you are well prepared, you truly have nothing to worry about. The examiners are mostly trying to help you, and they want to see you succeed. A lot of their advice may be useful for your research project.

Dissertation

To be filled in!

Here are very basic things you might be wondering in a Q&A format:

Thank you!

Thank you for contributing to this handbook!

- Saurabh Bajaj
- Alireza Faghaninia

- Joey Montoya
- John Dagdelen
- Nils Zimmerman
- Ben Ellis
- Maksim Rakitin
- Daniel Dopp
- Todd Karin
- Alex Dunn
- Viktoriia Baibakova
- Xin Chen
- Alex Ganose
- Sanghoon Lee
- Zhuoying Zhu